Санкт-Петербургский государственный университет

В.В. Монахов, О.В. Огинец, С.Н. Жоголь, М.Г. Яковлева

СОЗДАНИЕ ВИРТУАЛЬНЫХ ПРИБОРОВ И ПРОГРАММИРОВАНИЕ УСТРОЙСТВА СБОРА ДАННЫХ NI myDAQ В СРЕДЕ LABVIEW

Учебное пособие
2-е издание, переработанное и дополненное

Санкт-Петербург
2017

ББК 32.973.26-018.2

В311я43+з973.2-018я43

Рецензенты: доктор физ.-мат. наук, профессор С.Л.Яковлев,

старший преподаватель А.Г.Смирнов

Печатается по решению методической комиссии физического факультета СПбГУ

В.В. Монахов, О.В. Огинец, С.Н. Жоголь, М.Г. Яковлева

В311я43+з973.2-018я43 Создание виртуальных приборов и программирование устройства сбора данных NI myDAQ в среде LABVIEW**:** Учебное пособие. – 2-е изд., перераб. и доп. – СПб: ЛЕМА, 2017. – 131 с.: ил.

Учебное пособие содержит описания трех лабораторных работ, проводимых в лаборатории автоматизированного практикума по физике в рамках дисциплины «Физический практикум» для студентов 2 курса бакалавриата, обучающихся на образовательных программах «Физика», «Радиофизика», «Прикладные физика и математика» СПбГУ. Может быть полезно для студентов других вузов, изучающих автоматизацию физического эксперимента.

УДК 681.3.068
ББК 32.973.26-018.2

СОДЕРЖАНИЕ

I. Лабораторная работа № 1-NI. Создание виртуальных приборов в среде LabView

1. Теоретическая часть

1.1. Среда LabView. Виртуальные приборы и управление реальными приборами

1.1.1. Среда LabView и виртуальные приборы

LabView (от *Laboratory Virtual Instrument Engineering)* — среда разработки программ на графическом языке программирования «G» (от “Graphic” - графический) компании National Instruments (NI). Данная компания является мировым лидером в области автоматизации эксперимента – создает как аппаратуру автоматизации эксперимента, так и удобные средства разработки программного обеспечения, предназначенного для управления этой аппаратурой. Выпускаемое компанией NI оборудование позволяет создавать установки и управлять технологическими процессами в различных областях науки и техники. Большая часть программного обеспечения этих установок разрабатывается в среде LabView.

Графический интерфейс LabView достаточно прост для программирования, а создаваемое в среде LabView программное обеспечение обладает большой гибкостью, поскольку при изменении конфигурации установки позволяет очень просто менять программу управления этой установкой. А сами программы для небольших установок получаются простыми и понятными.

Программы, создаваемые в среде LabView, называются виртуальными приборами (Virtual Instruments, сокращенно – VI). Они сохраняются в файлах с расширением **.vi**. Компоненты, из которых строятся виртуальные приборы, сами также являются виртуальными приборами. Эти компоненты напоминают микросхемы со входами и выходами, а написание программы заключается в расположения компонентов на схеме, задании их внутренних параметров (числовых значений, текста и т.п.) и подсоединении

виртуальными проводами выходов к необходимым входам. Таким образом, написание программы напоминает составление электрической схемы (рис.1.1).

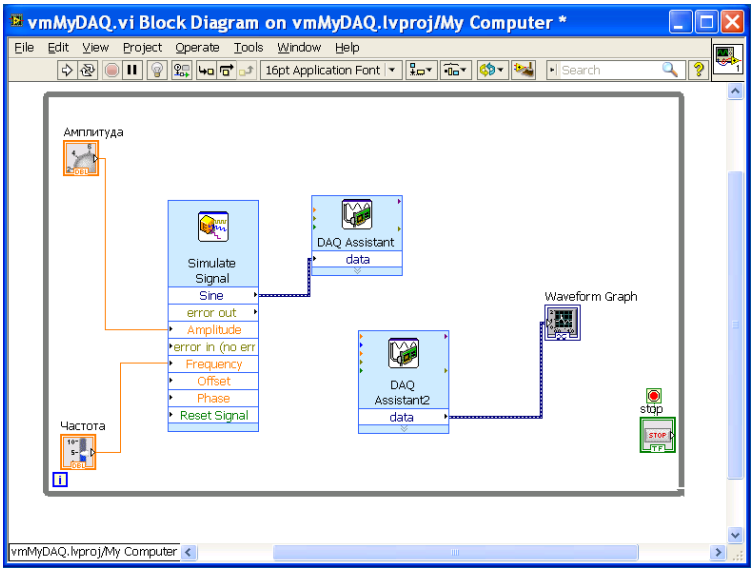

Рис.1.1. Пример программы управления измерительной установкой – программно управляемого генератора электрических сигналов заданной формы (левая часть схемы) и цифрового осциллографа (правая часть схемы).

- Алгоритмы управления установкой, не связанные виртуальными проводами на схеме, выполняются параллельно, и от программиста не требуется никаких усилий для такой параллелизации работы программы. Это связано с тем, что в программном обеспечении чаще всего используют модель вычислений, основанную на ***потоке управления*** (Controlflow), а LabView использует другую модель вычислений – ***потока данных*** (Dataflow). В ней программа представляется в виде графа. Узлы этого графа – элементы блок-схемы программы, являются аналогами операторов обычной программы, а стороны графа (связи на блок-схеме) показывают, от каких узлов данные поступают к другим. Очередной узел сразу начинает выполнение, как только становятся доступными все его

входные данные. Когда узел обработал данные, результаты появляются на его выходах и передаются всем узлам, с которыми соединены его выходы.

Аналогичная методика проектирования используется в пакете Simulink в MatLab, но следует учесть, что этот пакет в первую очередь ориентирован на моделирование процессов в электрических цепях и т.п., а LabView – **для автоматизации эксперимента, управления установками**. Хотя постепенно различия в областях применения Simulink и LabView уменьшаются – в последние годы в Simulink появились возможности управления устройствами, выпускаемыми NI, а LabView начали применять для математического моделирования.

Спроектированный из компонентов виртуальный прибор может быть оформлен в виде компонента со скрытым содержимым и выходящими наружу из его корпуса входами и выходами. В таком виде его можно использовать для расположения на блок-схеме при построении более сложных виртуальных приборов.

Название “виртуальные приборы” не означает, что с их помощью только имитируют реальные приборы и процессы, происходящие в лабораторных или измерительных системах. К каждому устройству производства National Instruments в LabView имеются компоненты, осуществляющие управление этим устройством. Например, компонент DAQ Assistant позволяет управлять цифровыми портами ввода-вывода, цифро-аналоговыми преобразователями (ЦАП) и аналогово-цифровыми преобразователями (АЦП) – управлению этими устройствами с помощью LabView будет посвящен ряд следующих лабораторных работ.

Таким образом, программы, написанные в среде LabView, при наличии соответствующих приборов, подсоединенных к компьютеру, позволяют генерировать и преобразовывать реальные физические сигналы, производить измерения, управлять внешними приборами и пересылать данные на другие компьютеры.

В этой и следующих работах студент познакомится с основами программирования в среде LabView, с основными принципами построения виртуальных приборов с помощью средств LabView, и выполнит ряд практических заданий, что позволит получить навыки в освоении этого современного инструмента исследования и управления физическими процессами.

1.1.2.Общие принципы построения программы в среде LabView. Виртуальный прибор

Виртуальный прибор состоит из следующих частей:

Лицевая панель (Front panel) является интерфейсом виртуального прибора и имитирует лицевую панель обычного измерительного прибора с его традиционными элементами управления *(Controls*). Это кнопки, тумблеры, индикаторы, ручки, измерительные шкалы, дисплеи и т.д. При помещении на лицевую панель какого-либо компонента на блок-схеме (см. далее) автоматически появляется соответствующая этому компоненту пиктограмма . Данные, вводимые с помощью элементов лицевой панели, обрабатываются с соответствии с алгоритмами, заданными блок-схемой. Результаты выполнения этих алгоритмов выводятся на элементы отображения информации на лицевой панели.

Блок-схема (Block diagram). В окне блок-схемы располагается программный код виртуального прибора, созданный на языке G. Компонентами блок-схемы могут быть пиктограммы элементов лицевой панели, виртуальные приборы более низкого уровня, чем разрабатываемый виртуальный прибор, встроенные функции LabView, константы, а также структуры управления выполнением программы. Также на блок-схему можно помещать компоненты, не отображающиеся на лицевой панели (например, компоненты арифметических действий и других алгоритмические конструкций). В окне блок-схемы можно задавать алгоритмы математических операций, объединять компоненты в виртуальный прибор, опрашивать порты ввода-вывода, производить другие операции. Связь между

различными элементами блок-схемы создается с помощью соединений, называемых виртуальными проводами (виртуальными проводниками).

1.1.3.Типы данных и вид виртуальных проводов

В LabView имеются несколько типов данных, каждому из них соответствует цвет и внешний вид проводов и внешней окантовки компонентов на блок-схеме, а также подписи в нижней части компонента на блок-схеме (см.Таблицу 1). Провода с аналоговыми сигналами показываются сплошной линией, с дискретными величинами (сигналами от выключателей, нажатия кнопок и т.д.) – пунктиром, строковыми данными – извилистой линией (“змейкой”), с массивами – более жирной линией соответствующего вида, с двумерными массивами – сдвоенной жирной линией соответствующего вида. Например, провод для сигнала с одномерным массивом целочисленных измерений будет показан в виде толстой синей линии (целочисленным данным соответствует синий цвет).

Таблица 1 .

Тип данных	**Варианты типа и обозначение типа на компоненте**	**Цвет провода и окантовки**
Логический (Boolean)	TF – сокращение от значений True/False, принимаемым булевскими переменными	Зелёный
Числа в формате с плавающей точкой (Floating point) – используются для хранения вещественных данных	EXT – Extended, расширенная точность (19 значащих десятичных цифр мантиссы, значения до 10^{4932}, а также бесконечность Inf и -Inf) DBL – Double, двойная точность (14 значащих десятичных цифр мантиссы, значения до 10^{308}, а также бесконечность Inf и -Inf) SGL – Single, одинарная точность (7 значащих десятичных цифр мантиссы, значения до 10^{38}, а также бесконечность Inf и -Inf)	Оранжевый
Комплексные числа в формате с плавающей точкой	CXT – Complex Extended, расширенная точность CDB – Complex Double, двойная точность CSG – Complex Single, одинарная точность	Оранжевый

Числа в формате с фиксированной точкой (Floating point) – используются для хранения вещественных данных	Можно задавать число бит, отводимых под целую и дробную части числа. Диапазон изменения гораздо меньше, чем для типов в формате с плавающей точкой – но можно обеспечить большее количество значащих цифр числа.	Серый
Целые числа	I64 – 64-разрядные целые со знаком. I32 – 32-разрядные целые со знаком. I16 – 16-разрядные целые со знаком. I8 – 8-разрядные целые со знаком. U64 – беззнаковые 64-разрядные целые. I32 – беззнаковые 32-разрядные целые. I16 – беззнаковые 16-разрядные целые. I8 – беззнаковые 8-разрядные целые.	Синий
Строка	abc – обозначено по первым буквам латинского алфавита.	Розовый
Массив	[] – квадратные скобки. Внутри указывается тип ячейки массива.	Соответствует типу ячейки массива
Осциллограмма (waveform)	Предназначен для обработки реальных сигналов. Содержит данные, начальное значение времени и интервал времени между измерениями.	Соответствует типу передаваемых данных
Кластер	Предназначен для объединения сигналов разных типов для прохождения по одному проводу. Используется для уменьшения числа проводов на схеме.	Коричневый провод, внутри – пунктир из белых точек
Динамические типы	Предназначены для передачи данных, генерируемых виртуальными приборами (обычно идущих через заданные интервалы времени).	Темно-синий провод, внутри – пунктир из белых точек

1.2. Создание нового проекта. Окна проекта

После запуска LabView открывается окно Getting Started. В стартовом окне (рис.1.2) в среде LabView 2013 или 2012 выбирается кнопка Create Project. В более ранних версиях данная кнопка отсутствует, и следует выбрать пункт Blank VI.

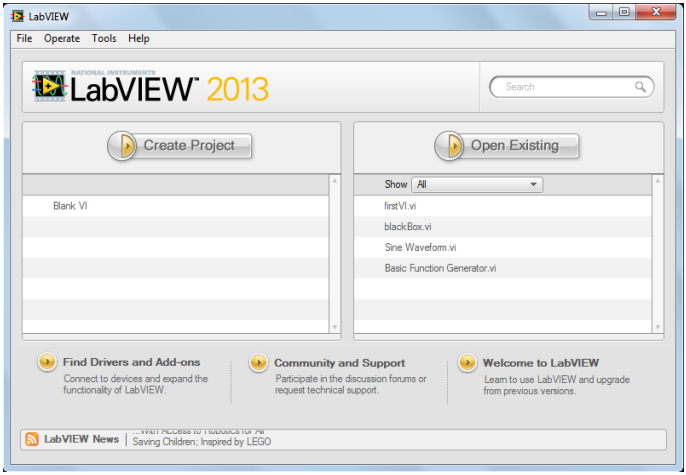

Рис.1.2. Стартовая страница.

В LabView 2013 в появившемся окне (рис.1.3) необходимо создать проект Blank VI (пустой виртуальный инструмент), для этого по соответствующей строчке необходимо сделать двойной щелчок. В более старых версиях этот этап не нужен – сразу будет создан проект Blank VI.

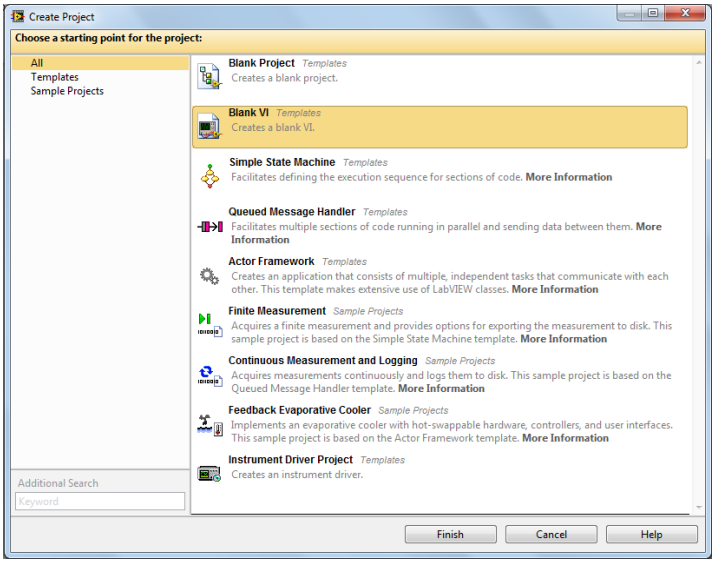

Рис.1.3. Страница создания проекта.

После этого в рабочей области среды должны появиться два окна (рис.1.4), *Front Panel* (лицевая панель) и *Block Diagram* (блок-схема).

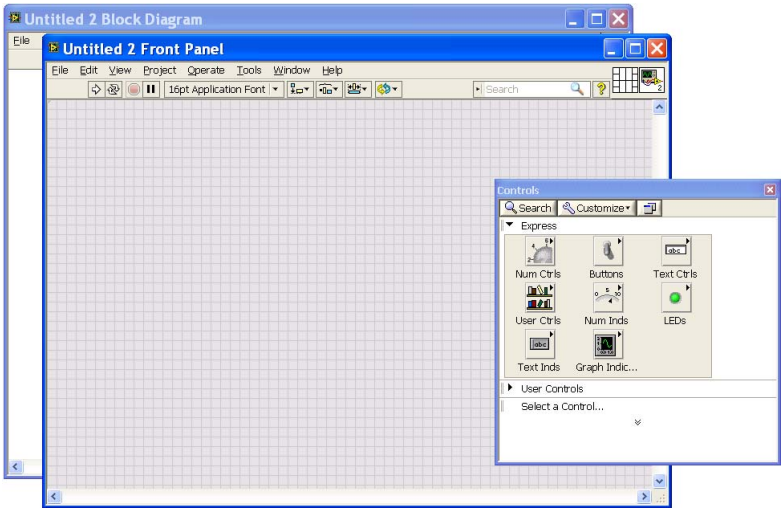

Рис.1.4. Вид рабочей области, на переднем плане окно ***Front Panel*** .

При переходе в окно *Front Panel* показывается окно *Controls* (рис.1.4), а при переходе в окно *Block Diagram* – окно *Functions* (рис.1.5)

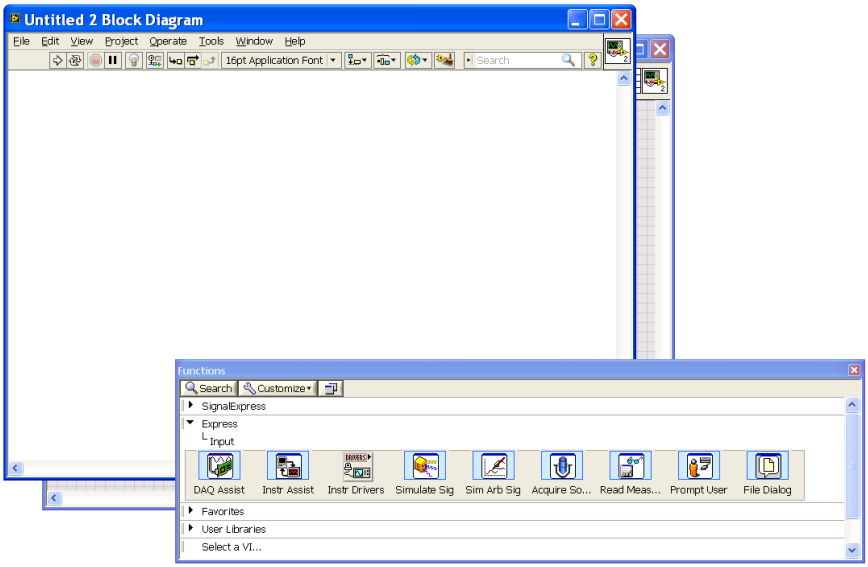

Рис.1.5. Вид рабочей области, на переднем плане окно ***Block Diagram***.

В случае если при переходе в окно *Front Panel* не отображается окно *Controls* (вы его закрыли), в верхнем меню окна *Front Panel* следует перейти в пункт меню View и выбрать пункт Controls Palette. Если не отображается окно *Block Diagram* (вы его закрыли), в окне *Front Panel* следует перейти в пункт меню Window и выбрать пункт *Block Diagram*.

1.3.Библиотеки виртуальных приборов

Элементы управления, функциональные узлы и ВП вызываются с помощью иерархической системы меню - палитр компонентов *Controls, Functions, Tools* и т.д. Каждая из палитр доступна только если открыта соответствующая панель. За каждым разделом палитры закреплен набор средств управления и индикации (для *Front Panel*) или набор функций и алгоритмов (для *Block Diagram).*

LabView обладает развитой библиотекой функций и подпрограмм, реализующих большое число типичных задач программирования. Алгоритм работы виртуального прибора формируется с помощью палитры *Functions*, отображаемой при переходе в окно *Block Diagram* (рис.1.6). Если оно не отображается, то в меню окна *Block Diagram* следует зайти в пункт View и выбрать пункт *Functions*.

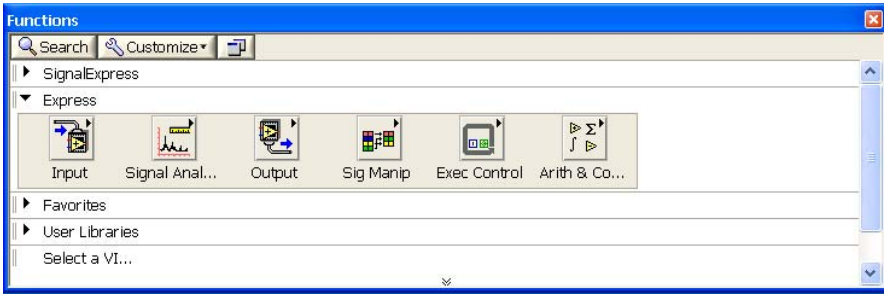

Рис.1.6. Вид окна Functions.

В окне *Functions* во вкладке *Express* расположены подразделы библиотеки *Express* виртуальных приборов. На пиктограммах подразделы помечаются черным треугольником в правом верхнем углу пиктограммы (рис.1.6). В

подразделе *Express/Input* располагаются приборы, позволяющие принимать информацию с внешних устройств, а также открывать диалоговое окно выбора файлов (компонент *FileDialog*) (рис.1.7).

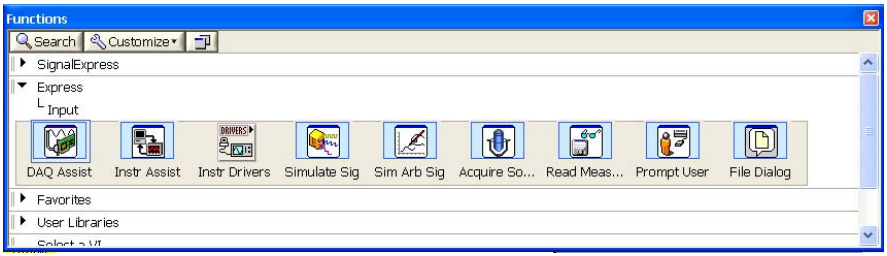

Рис.1.7. Вид раздела *Express/Input окна Functions*.

В разделе *Express/Output* находятся приборы, позволяющие отправлять информацию на внешние устройства (рис.1.8).

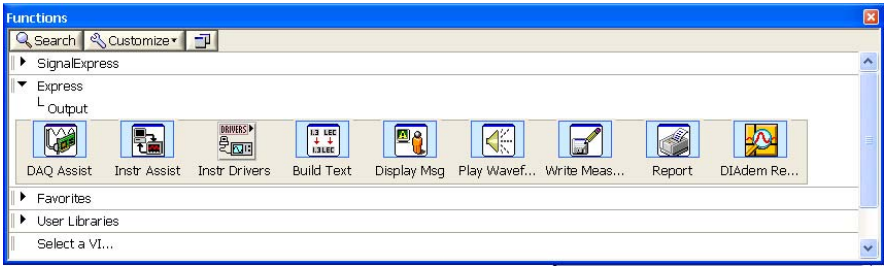

Рис.1.8. Вид раздела *Express/Output окна Functions*.

В *Express/Signal Analysis* располагаются приборы, с помощью которых можно анализировать сигналы, а также их синтезировать (рис.1.9).

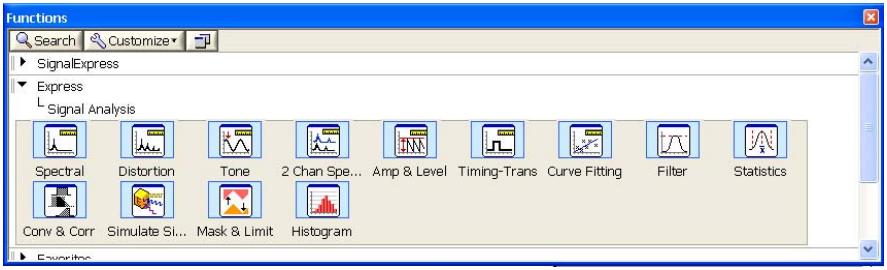

Рис.1.9. Вид раздела *Express/ Signal Analysis окна Functions.*

Для выбора приборов преобразования сигналов служит раздел *Express/Signal Manipulation* (рис.1.10).

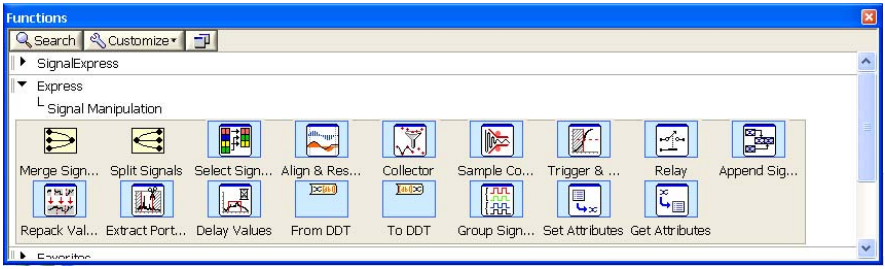

Рис.1.10. Вид раздела *Express/ Signal Manipulation окна Functions*.

Приборы, обеспечивающие работу структур программирования *while..do* (цикл), *case* (условие выбора), *Flat sequence* (обеспечение выполнения действий в заданном порядке), *Time delay* (задание задержки по времени) и *Elapsed Time* (измерение интервалов времени), расположены во вкладке *Express/Execution Control* (рис.1.11).

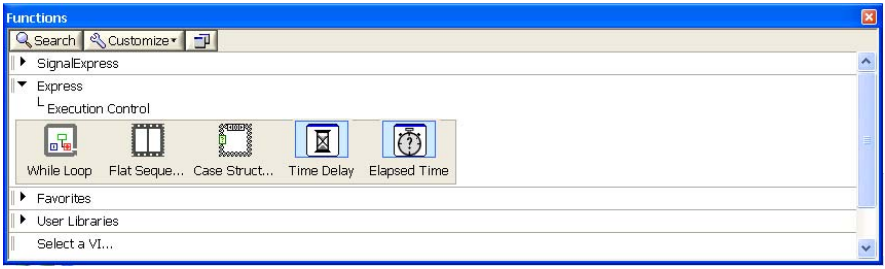

Рис.1.11. Вид раздела *Express/ Execution Control окна Functions*.

Приборы, обеспечивающие выполнение алгебраических и математических операций, расположены в *Express/Arithmetic&Comparison* (рис.1.12).

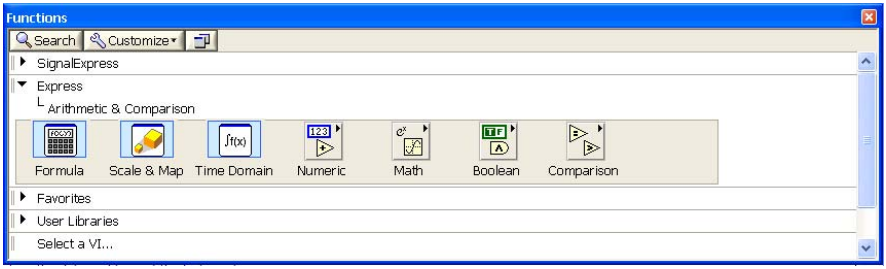

Рис.1.12. Вид раздела *Express/Arithmetic&Comparison*.

В этом разделе содержатся приборы *Formula, Scale&Map, Time Domain,* а также подразделы (напомним, что на пиктограммах подразделы помечаются

черным треугольником в верхнем правом углу пиктограммы): *Numeric, Math*, *Boolean, Comparison.*

Справку (на английском языке) по разделу или виртуальному прибору можно найти, нажав клавишу F1 и введя в строку поиска имя раздела или прибора.

***Замечание 1*:** в раздел *Functions/Express* для окна *Block Diagram* и раздел *Controls/Express* для окна *Front Panel* вынесены только наиболее часто используемые компоненты (виртуальные приборы). Более полный набор математических операций и циклов можно найти в разделе *Functions/Programming*, а элементов управления (в том числе для работы с таблицами строк, массивами и др.) - в разделах *Controls/Modern, Controls/Silver, Controls/System* и др. Например, элементы управления (*Controls)* для работы с булевскими данными (кнопки, переключатели, виртуальные светодиоды) можно найти в разделе *Controls/Modern/Boolean*.

Замечание 2: добавление новых элементов схемы удобно проводить с помощью меню, появляющегося по щелчку правой кнопкой мыши.

1.4. Некоторые элементы программирования LabView, использующиеся в лабораторных работах

1.4.1. Работа с виртуальными проводами

Соединения с помощью виртуальных проводов должны выполняться в соответствии с рядом правил.

1) Подключение компонентов

Каждый провод можно подсоединять к одному (и только одному) источнику сигнала – т.е. к выходу компонента. Однако каждый провод можно подсоединять к произвольному количеству входов компонентов (например, входов индикаторов) – этот способ называется разветвлением. Так, никогда нельзя соединять вместе выходы элементов управления, но можно присоединить выход элемента управления к входам двух или более индикаторов.

Входы у виртуальных приборов могут быть обязательными и необязательными для подключения. Не должно быть компонентов, содержащих не подсоединенные входы, обязательные для подключения. Например, у компонентов арифметических действий все входы являются

обязательными для подключения. В таких случаях попытка запустить программу с неподключенным входом вызывает сообщение об ошибке.

2) Соединение проводами

При проведении соединений следует учесть, что если провод отпускается в тот момент, когда его конец находится над другим проводом, образуется соединение проводов, обозначаемое точкой в месте соединения. Если провода пересекаются, но соединительной точки нет, то считается, что они проходят один над другим, и соединения проводов нет.

За провод можно схватиться мышью и перетащить его в сторону, при этом внешне он может поменять место подсоединения к компонентам (например, точка подключения может с передней части компонента переместиться на его боковую сторону), но этого не следует бояться – провод останется присоединен к тем же выходам и входам компонентов.

Линейный участок провода можно выделить щелчком левой кнопки мыши (провод выделяется курсивом) и перетащить в удобное место, либо удалить. При двойном щелчке выделяется весь провод.

Если вы начали тащить провод и хотите в некоторой точке изменить на 90º его дальнейшее направление, отпустите кнопку мыши в этой точке и тяните провод дальше. В новой точке, где необходим поворот провода, щёлкните кнопкой мыши, и т.д.

3) Отмена и удаление соединений

Если вы начали тащить провод, но не закончили, отпустите кнопку мыши и нажмите на клавиатуре кнопку Esc – и пунктирная заготовка провода пропадёт.

Для удаления провода его необходимо вручную выделить, дважды щёлкнув по нему, и нажать клавишу Delete.

Если вы осуществили ошибочное соединение, перемещение или удаление проводов, можно вернуться к предыдущему состоянию нажатием комбинации клавиш Ctrl+Z или через меню Edit/Undo.

4) Разорванные провода

Существует понятие поврежденного (разорванного) провода, который не соединяет должным образом источник сигнала с входом приемника сигнала. Разорванный провод выглядит как пунктирная линия с красным крестом посредине. Наличие такого провода не позволяет запустить программу виртуального прибора на выполнение – выведется сообщение об ошибке.

Разрыв проводов может происходить по различным причинам – например, при соединении двух объектов с различными или несовместимыми типами данных. Так, нельзя подключать выход логического элемента управления к числовому входу, числовой выход к входу логического элемента управления, соединять выходы, и т.д.

Для быстрого удаления всех поврежденных проводов следует использовать «горячую» комбинацию клавиш Ctrl+B.

5) Автоматическое расположение проводов

Запутанное расположение проводов на блок-схеме часто бывает неудобным, и и LabVIEW предоставляет возможность оптимизировать их размещение для большей наглядности .Для этого нужно выделить неудачно расположенные провода, щелкнув по ним правой кнопкой мыши при нажатой клавише Shift, после чего воспользоваться комбинацией клавиш Ctrl+U (соответствует пункту меню *Edit/Clean up Selection).* Для оптимизации размещения всех элементов блок-схемы следует воспользоваться комбинацией клавиш Ctrl+U (соответствует пункту меню *Edit/Clean up Diagram*) без выделения какого-либо элемента.

Автоматическое расположение не всегда приводит к удачным результатам, и можно вернуться к предыдущему состоянию нажатием комбинации клавиш Ctrl+Z или через меню *Edit/Undo*. После чего можно передвинуть часть проводов и повторить попытку – при другой начальной конфигурации результат автоматического размещения может оказаться более удачным.

1.4.2. Возможности формирования лицевой панели

Цифровые элементы ввода и вывода на лицевой панели могут иметь разный вид. Так, например, индикаторы из группы *Numeric* палитры могут быть

просто цифровыми (*Num Ind*), скользящими (*Slide*), вращательными (*Knob*, *Dial)* и т.д. Размер окна или шкалы можно увеличить, поместив курсор мыши на любой из крайних углов элемента и потянув за него.

В лабораторной работе №3 в качестве элементов управления выходным сигналом порта вывода будут использоваться тумблеры *Toggle Switch (Front panel/Controls/Express/Buttons&Switches/Toggle Switch).* Эти элементы аналогичны переменным типа Boolean в языках высокого уровня и имеют два значения – *True* (1) или *False* (0). На панели *Block Diagram* эти элементы имеют рамку зеленого цвета, а соединительные провода изображаются зеленой пунктирной линией.

LabView позволяет делать надписи на элементах, менять диапазон шкал, добавлять комментарии, что позволяет сделать лицевую панель ВП наиболее удобной для работы. Для ввода надписи нужно дважды щелкнуть по иконке элемента или правой кнопкой мыши вызвать меню и выбрать раздел *Properties* (Свойства).

1.4.3. Запуск программы

Запускать программу можно и из окна лицевой панели, и из окна блок-схемы с помощью кнопок. Стрелка (*Run*) - однократный запуск алгоритма программы, двойная стрелка (*Continuously Run*) – циклический запуск, означающий, что после окончания работы алгоритма программа запускается снова.

Полностью эквивалентен запуску с помощью кнопки *Run* запуск из пункта меню *Operate/ Run*… либо комбинацией клавиш Ctrl+R.

Если программа содержит ошибку, то стрелка примет “ сломанный» вид. Если щелкнуть по «сломанной» стрелке, появится окно с информацией о выявленных ошибках (*Error List*), которые необходимо исправить и снова запустить программу.

1.4.4. Структуры – циклы и проверка условий

Выполнение работы программы в определенном порядке или повторение отдельных ее частей в зависимости от какого-либо условия осуществляется с помощью функций *Structures* (Структур), которые находятся в палитре *Functions/ Programming/ Structures* (рис.1.13).

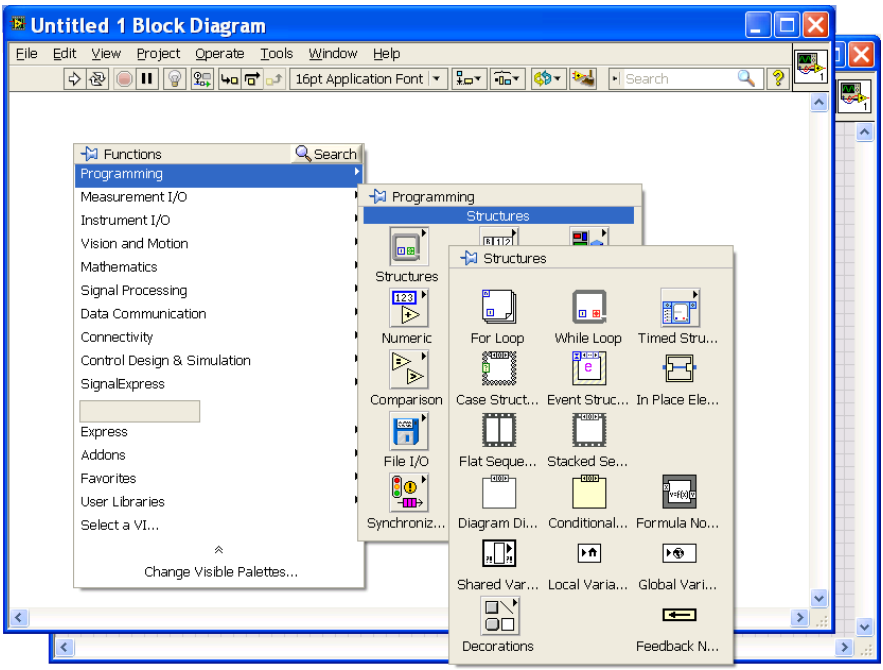

Рис.1.13. Вид раздела *Programming/ Structures* окна *Functions* при вызове меню правой кнопкой мыши.

Любая структура имеет вид рамки, по краям которой размещаются входной и выходной терминалы (ограничители). На рис.1.14 приведен результат перетаскивания структуры *While Loop* в окно блок-схемы.

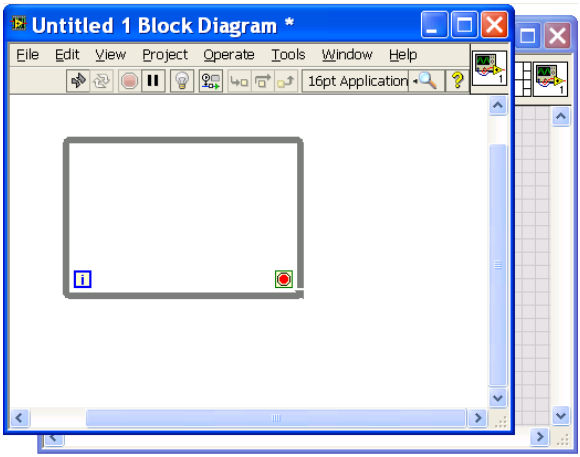

Рис.1.14. Результат перетаскивания структуры *While Loop* в окно блок-схемы.

При перетаскивании структуры в окно блок-схемы ее можно наложить на существующий участок программы или, наоборот, сначала расположить структуру в окне, а затем помещать внутрь рамки элементы программы.

Замечание 1: если структура накладывается на уже существующую программу, то если не все элементы блок-схемы поместятся внутрь ее рамки, они не будут помещены в структуру, и она будет просто перекрывать их видимость. Поэтому перемещать структуру нужно внимательно и осторожно.

Замечание 2: для непрерывной работы программы используется структура *While Loop* из раздела *Express/Execution Control/*, которая дополнительно к структуре *Functions/ Programming/ Structures/ While Loop* содержит кнопку *STOP* – см. раздел 2.2.

При выполнении лабораторных работ используются лишь некоторые из структур, имеющихся в библиотеке LabView, например, цикл по условию (*While Loop),* цикл с фиксированным числом итераций (*For Loop*), структура варианта выбора (*Case Structure*).

Цикл *While Loop* (по условию) работает до тех пор, пока будет выполняться логическое условие продолжения цикла.

В левом нижнем и правом нижнем углу рамки цикла, соответственно (рис.1.14), располагаются:

[i] – терминал счетчика итераций, содержащий номер текущей итерации, начиная с нуля.

– терминал условия продолжения *Loop Condition*. Цикл будет выполняться до поступления на терминал значения *False*.

Цикл *For Loop* (рис.1.15) выполняет участок программы, расположенный внутри рамки цикла определенное число раз. В левом верхнем и нижнем углах рамки, соответственно, располагаются:

[N] – терминал общего числа итераций,

[i] – терминал счетчика итераций, содержащий номер текущей итерации, начиная с нуля.

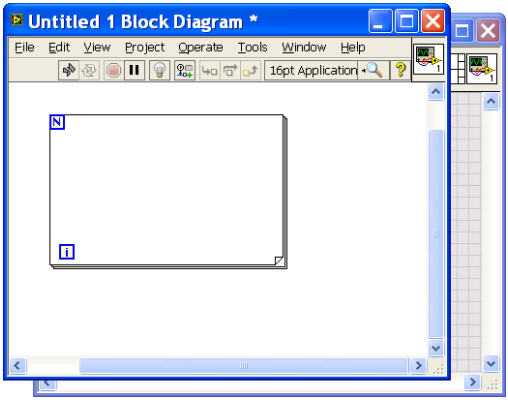

Рис.1.15. Цикл *For Loop*.

Структура *Case Structure* (рис.1.16) может иметь внутри себя одну или более вкладок-условий, которые работают при выполнении заданных пользователем условий. По умолчанию *Case Structure* появляется на блок-схеме с вкладками-заготовками для двух условий – *True* и *False*. Переключиться между вкладками можно щелчком по стрелкам около значений *True* и *False*.

Небольшой прямоугольник , находящийся посередине левой стороны рамки – терминал селектора данных. Он управляет тем, алгоритм какой вкладки будет использован. Если на него подается значение *True*, будет выполняться алгоритм из вкладки *True*, иначе – алгоритм из вкладки *False*.

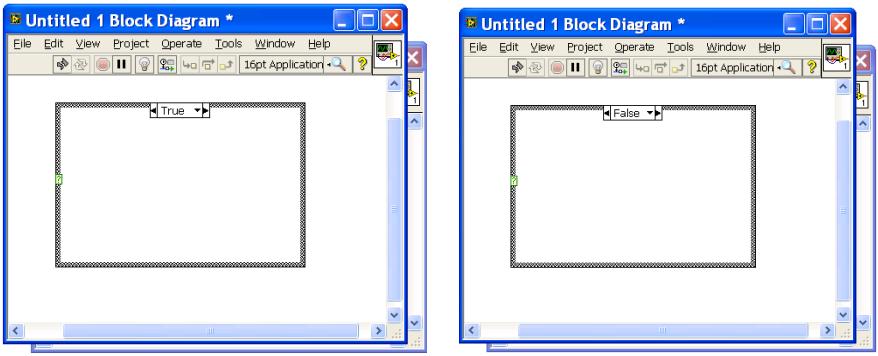

Рис.1.16. *Case Structure* – переключение между вкладками *True* и *False* производится нажатием на стрелку около подписей *True* и *False*.

В случае поступления на вход условия значения *True* на выход структуры будет поступать результат выполнения алгоритма, расположенного внутри вкладки *True*, а при поступлении на вход условия значения *False* на выход будет поступать результат выполнения алгоритма, расположенного внутри вкладки *False*.

Важно: если требуется ввести значение из внешней части программы внутрь структуры, провод с входными данными следует подсоединить к рамке структуры в любом свободном месте (обычно это делают с левой части рамки). После этого на рамке возникает терминал в виде квадратика, от которого можно тянуть провода к элементам внутри структуры.

Если же требуется вывести данные из структуры, провод с выходными данными, полученными от элемента, находящегося внутри структуры, следует подсоединить к рамке структуры в любом свободном месте (обычно это делают с правой части рамки). После этого на рамке возникает терминал в виде квадратика, от которого можно тянуть провода к наружным элементам.

Для *Case Structure* подобные действия необходимо проделать для каждой из вкладок, при этом входные и выходные терминалы, созданные в одной из вкладок, доступны во всех вкладках. Входной терминал можно не использовать, если в конкретном варианте поступающие на него данные не

нужны, а на выходной терминал требуется подавать данные в каждой из вкладок.

На рис.1.17 в случае поступления на вход условия значения *True* на выход подается значение с компонента *Knob* – внутри вкладки *True* провод идет со входа на выход, т.е. что приходит на вход, то подается на выход. А если условие *False*, на выходе структуры появится значение -1, т.к. внутри вкладки *False* на выход поступает значение с числовой константы -1.

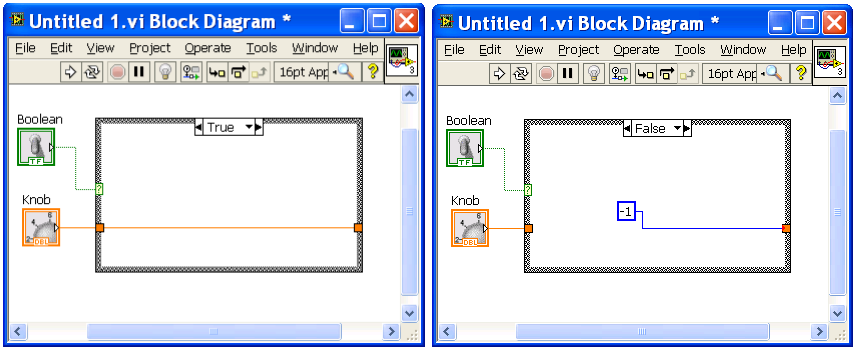

Рис.1.17. Пример ввода и вывода значений во вкладках *Case Structure*.

1.5. Компиляция программы в exe-файл

Программы, написанные с помощью LabView, могут быть скомпилированы и перенесены на другой компьютер.

Получившийся в результате компиляции exe-файл запускается как обычная программа Windows, и не требует наличия на компьютере коммерческой версии LabView. Однако для работы файла на компьютере необходимо установить исполняющую среду LabView из свободно распространяемого файла (для LabView 2011 SP1 это LVRTE2011SP1f1std.exe, для LabView 2012 – LVRTE2012f3std.exe, и т.д.). Учтите, что размер файла более 200 Мб. Кроме того, для запуска данной программы на другом компьютере необходимо скопировать на этот компьютер все файлы из папки, в которой создан exe-файл, в том числе вложенную папку data. Если переносить только exe-файл, он не будет запускаться.

1.6. Компьютерный тест №1

В тесте проверяется усвоение учащимся теоретической части описания лабораторной работы. Для выполнения теста следует зайти на сайт http://distolymp2.spbu.ru/www/olymp/ под своим логином (можно использовать тот же логин что и на практикуме по информатике на 1 курсе, если на занятиях использовалась система distolymp2) и перейти по ссылке “Компьютерный тест №1”.

Если логин отсутствует, следует зарегистрироваться по адресу http://distolymp2.spbu.ru/www/olymp/registration/autoreg/ .

После прохождения теста в разделе «Результаты» правильные ответы помечаются зеленым цветом, неправильные – красным. Результаты проверяются преподавателем и служат допуском к выполнению лабораторной работы.

2. Демонстрационные примеры

2.1. Демонстрационный пример 1 – виртуальный калькулятор, выполняющий арифметические действия

Рассмотрим создание нового виртуального инструмента (прибора) на примере показа результатов арифметических действий. Как уже говорилось, написание программы в LabView напоминает проектирование электрической схемы. Поэтому о передаваемых и обрабатываемых данных можно говорить как о сигналах. В частности, для двух сигналов можно говорить об их сложении, вычитании, умножении и даже делении.

Чтобы ознакомиться со структурой LabView и понять логику ее работы, рассмотрим пример сложения и умножения двух сигналов A и B, поступающих от перемещаемого пользователем ползунка и поворачиваемой пользователем ручки. Для решения этой задачи в окне *Front Panel* нужно построить лицевую панель виртуального прибора, а в окне *Block Diagram* блок-схему с алгоритмом действия прибора.

Сначала на лицевую панель надо установить числовые элементы A и B, от которых мы будем получать числовые сигналы. Выберем в качестве них два

компонента из раздела *Controls/ Express/ Numerical Controls*. Для этого в окне *Controls* перейдем в раздел *Express/Num Controls* (рис.1.18).

Замечание: раздел *Express* имеется как в окне *Controls*, появляющемся при переходе в окно *Block Diagram*, так и в окне *Functions*, появляющемся при переходе в окно *Front Panel*. Однако содержимое этих разделов разное, и раздел *Num Controls* невозможно найти в окне *Functions/ Express*.

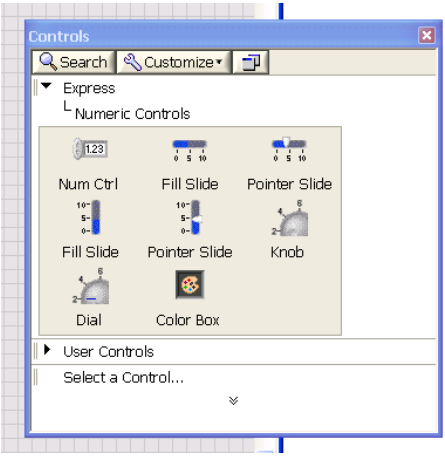

Рис.1.18. Вид раздела *Express/ Numerical Controls окна Controls.*

Перетащим на лицевую панель пиктограмму *Pointer Slide* (ползунок с указателем) и *Knob* (ручка) – на ней покажутся лицевые изображения этих компонентов (рис.1.19).

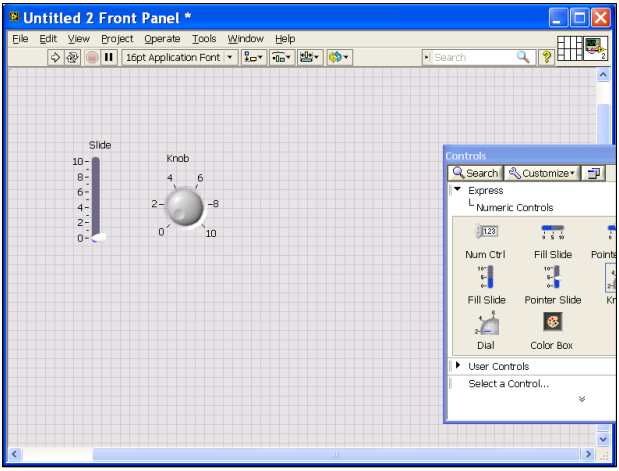

Рис.1.19. Вид окна *Front Panel* после перетаскивания в него компонентов *Pointer Slide и Knob*.

При этом на блок-схеме появятся пиктограммы данных приборов, с обозначениями входов и выходов (рис.1.20).

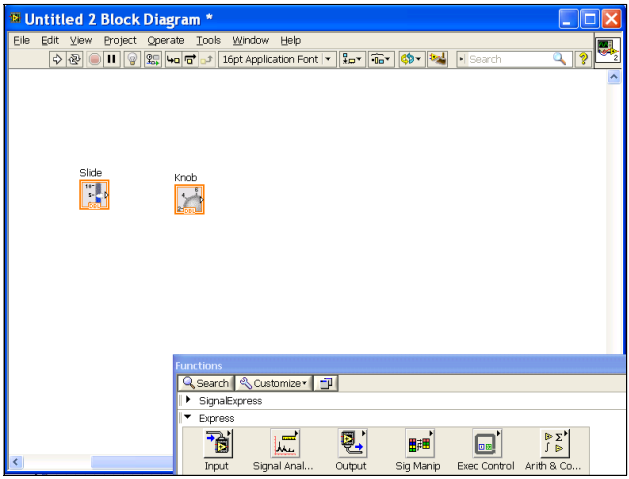

Рис.1.20. Блок-схема после перетаскивания на лицевую панель компонентов *Pointer Slide и Knob*.

Переместим на блок-схеме пиктограммы так, как нам удобно, а из окна *Functions/Express/Arith&Comparison/Numeric* перетащим в это окно компоненты *Add и Multiply* (рис.1.21).

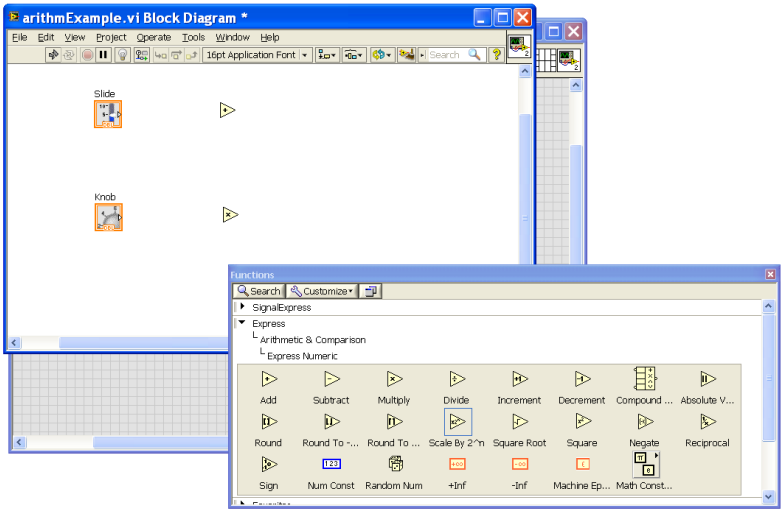

Рис.1.21. Блок-схема после перетаскивания на неё компонентов *Add* и *Multiply*.

Переключимся в окно *Front Panel* и перетащим на него из окна *Controls/Express/Num Inds* (сокращение от *Numeric Indicators* – числовые индикаторы) два компонента Num Ind (сокращение от *Numeric Indicator*) для показа значений A+B и A*B.

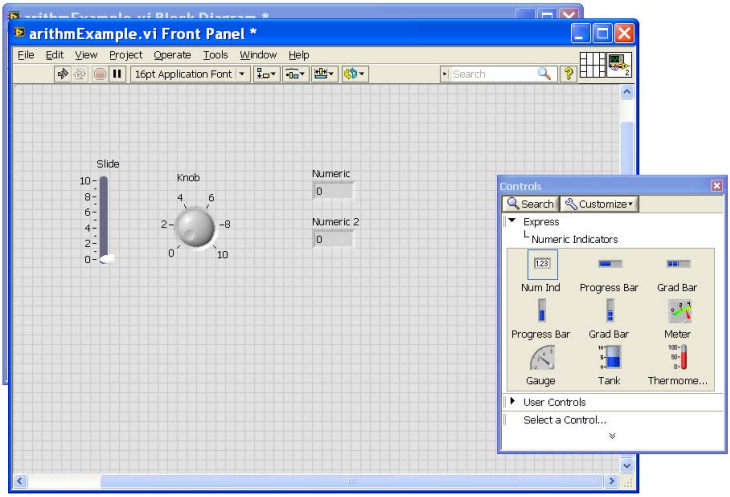

Рис.1.22. Лицевая панель после перетаскивания на неё компонентов *Num Ind*.

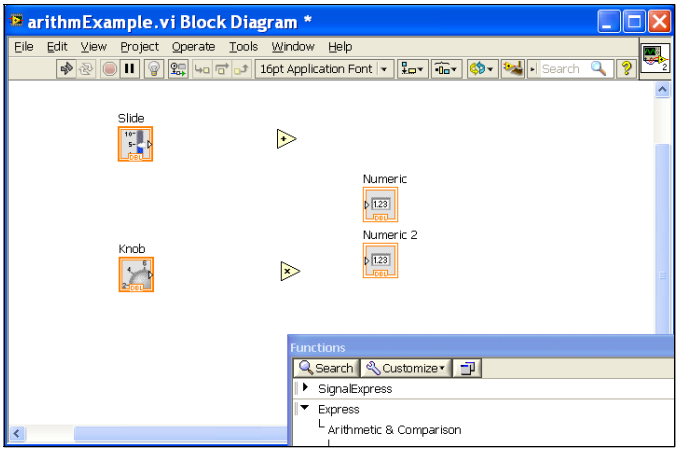

Рис.1.23. Блок-схема после перетаскивания на лицевую панель компонентов *Num Ind*.

Вид окон после этого показан на рис.1.22 и рис.1.23.

Для показа числовых значений в другой форме (полосы, шкалы стрелочного прибора и т.д.) служат другие компоненты числовых индикаторов. Подключим на блок-схеме выходы компонентов *Slide* и *Knob*, сигналы на которых соответствуют положению ползунка и угла поворота ручки, к входам компонента "+", а выход компонента "+" подсоединим к входу первого компонента *Num Ind* (на блок-схеме он обозначен как *Numeric*) – рис.1.24.

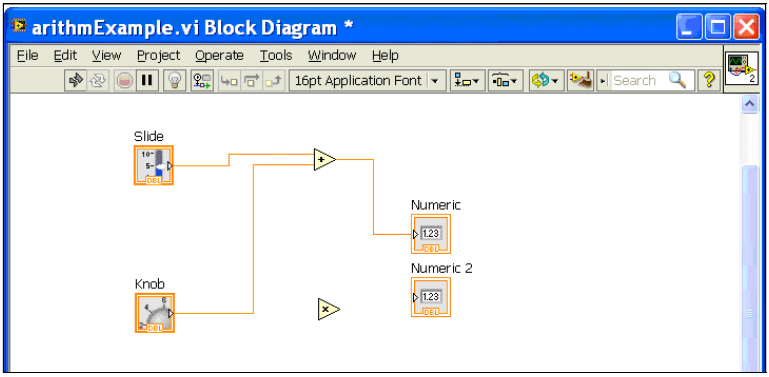

Рис.1.24. Блок-схема после соединения выходов и входов для показа результатов суммирования.

Подключение производится следующим образом. У виртуальных инструментов имеются контакты, помеченные треугольниками-стрелками. У входных контактов стрелки направлены внутрь прямоугольника, изображающего прибор, а у выходных – наружу. Для того чтобы подать сигнал с выхода одного виртуального прибора на вход другого, необходимо схватиться мышью за выходной контакт одного компонента и потащить появившуюся соединительную линию к входному контакту другого компонента. С таким же успехом можно начинать подсоединение провода от входа компонента к тому выходу, с которого на него подаётся сигнал.

Аналогичным образом соединим на блок-схеме выходы компонентов *Slide* и *Knob* с входами компонента умножения, а его выход – с входом индикатора *Numeric2* (рис.1.25).

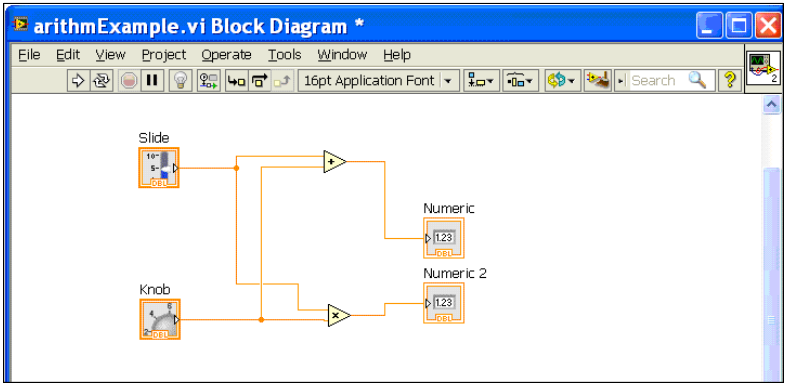

Рис.1.25. Блок-схема после проведения всех необходимых соединений.

Если вы сделали на лицевой панели или блок-схеме какие-то изменения и не сохранили изменённый проект в файле, в конце заголовка окна после имени проекта и слов “*Front Panel*” или “*Block Diagram*” будет виден символ * (рис.1.19-1.25). Не забывайте регулярно сохранять в файл результаты своего проектирования, поскольку иногда система LabView может оказываться неустойчивой и выходить из нормального режима работы, или может произойти сбой работы компьютера – и все результаты вашей работы пропадут.

Для того чтобы проверить работу созданного виртуального прибора, сначала следует сохранить его в файл с помощью меню File/Save. Если файл до этого не сохранялся, открывается диалоговое окно, в котором необходимо ввести имя сохраняемого файла. В данном примере для сохранения мы указали в качестве имени файла прибора arithmExample.

После этого надо перейти в окно *Front Panel* и нажать на иконку  на верхней панели (*Continuously Run*) – результат показан на рис.1.26. Мешающее окно *Controls*, часть которого показано в правой нижней части рис.1.26, можно закрыть.

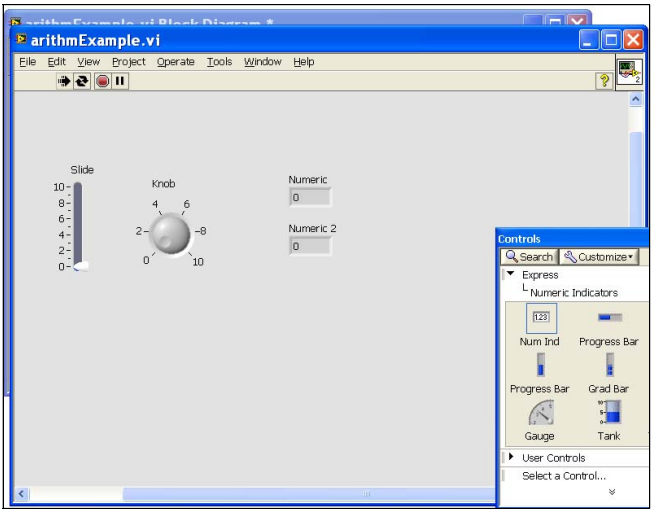

Рис.1.26. Работа программы – состояние после запуска.

Может показаться, что просто исчезла сетка привязки размещения компонентов – но на деле мы видим лицевую панель в окне работающей программы, и передвинуть компоненты на другое место не удастся. А при передвижении ползунка и повороте ручки на индикаторах показываются числовые значения арифметических действий (рис.1.27).

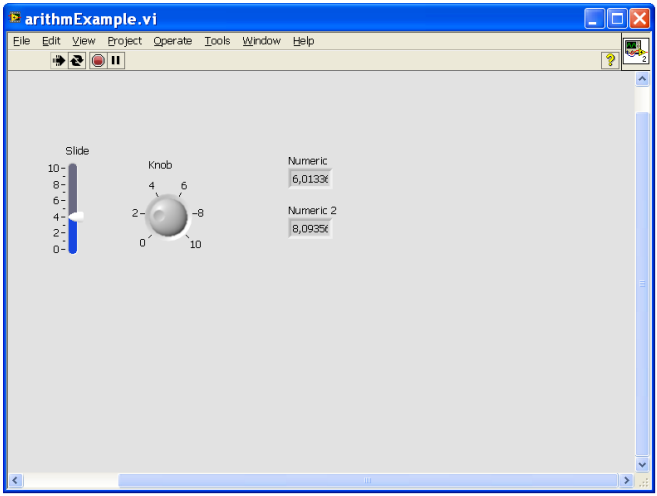

Рис.1.27. Работа программы – состояние после сдвига ползунка и поворота ручки.

Остановить работу программы можно нажатием кнопки  .

В приведенном примере результаты вычислений показывались в виде чисел. Чтобы ознакомиться с представлением результатов вычислений в другой форме, перейдите в окно *Front Panel* и перетащите на лицевую панель из окна *Controls/Express/Num Inds* компоненты *Progress Bar, Meter*, *Gauge*. (Не забывайте, что визуальные компоненты управления (*Controls*) можно добавлять только в окне *Front Panel*). Подсоедините на блок-схеме входы этих индикаторов к выходу компонента сложения.

Вы можете изменять параметры компонента, находящегося на лицевой панели. Для этого щелкните правой кнопкой мыши по компоненту, зайдите в раздел *Properties* (Свойства) и выполните необходимые настройки параметров.

2.2. Демонстрационный пример 2 – создание виртуального генератора и виртуального осциллографа

Рассмотрим еще один пример – создание прибора, совмещающего генератор сигналов и осциллограф, на экране которого показывается форма этих сигналов.

Создайте новый проект и перетащите на *Front Panel* иконку компонента графического дисплея *Waveform Graph* (окна построения графиков) – он находится в окне *Controls* (меню *Modern/Graph/Waveform/Graph*). Аналогичным образом устанавливается ручка регулировки амплитуды сигнала (*Controls/Modern/Numeric/Knob*). После произведенных операций получается картина, показанная на рис.1.28.

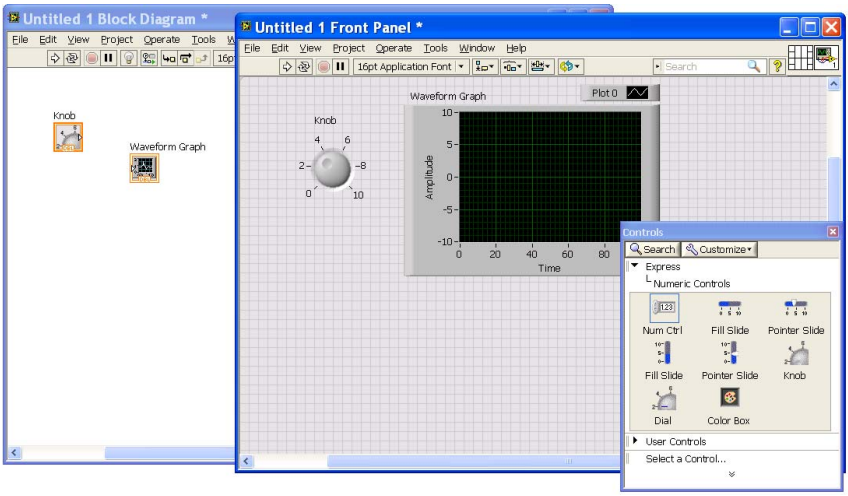

Рис.1.28. Создание интерфейса прибора.

Для имитации генератора сигналов в окне *Block Diagram* в разделе *Functions* необходимо выбрать компонент *Express/Signal Analysis/ Simulate Signal* и перетащить его иконку на панель *Block Diagram*.

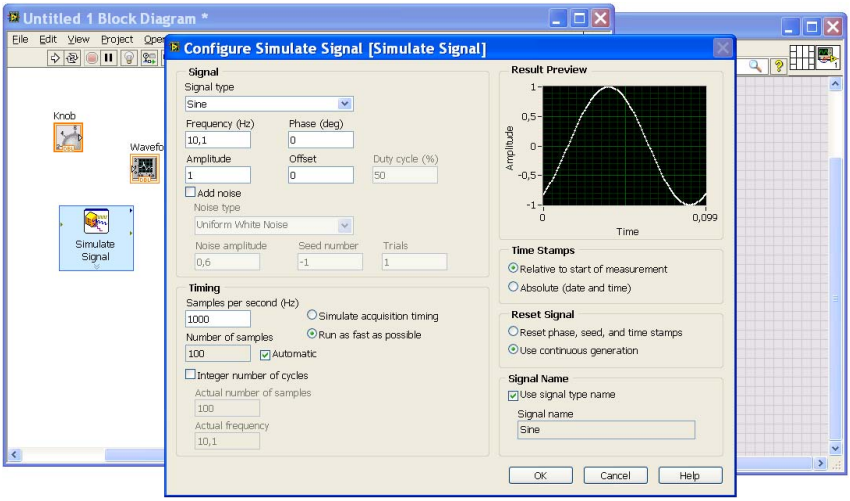

Рис.1.29. Создание виртуального генератора сигналов.

При этом сразу появится окно задания свойств генерации сигнала (рис.1.29). В дальнейшем для задания параметров сигнала необходимо сделать двойной щелчок по иконке генератора либо правой кнопкой мыши вызвать меню и выбрать пункт *Properties* (свойства). В открывшемся окне можно задать ряд параметров сигнала.

Справку по виртуальному прибору можно получить нажатием на кнопку Help в окне свойств прибора. Если разработчик прибора не назначил ему справку, открывается общая справка LabView.

Для того чтобы подать сигнал с генератора на осциллограф, необходимо схватиться мышью за выходной контакт сигнала генератора и потащить появившуюся соединительную линию к входному контакту компонента *Waveform Graph*, т.е. виртуального осциллографа (рис.1.30).

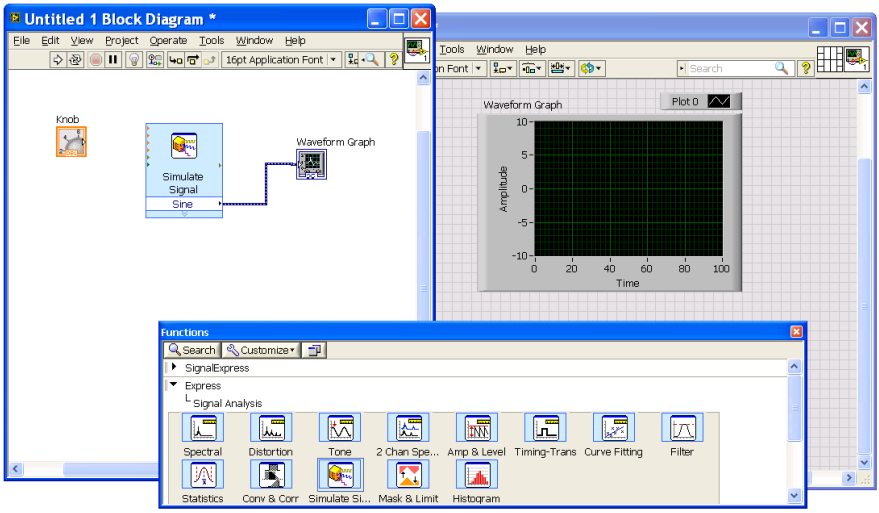

Рис.1.30. Подключение выхода генератора сигнала к виртуальному осциллографу.

Иногда для построения графиков, особенно на длительных промежутках времени, бывает удобнее использовать компонент *Waveform Chart* – в нём по оси времени откладывается текущее время, и идет постоянное обновление границ по истечении заданного промежутка времени. А в Waveform Graph по оси времени откладывается время, прошедшее с начала очередной порции данных, поэтому по времени начало графика соответствует 0, а окончание – размеру заданного интервала. Так, по умолчанию (см. рис.1.29) частота дискретизации сигнала генератора 1000 Гц (частота, с которой следуют точки измерения), и одна порция данных составляет 100 точек измерений (samples). Поэтому длительность порции данных, выводимых на экран цифрового осциллографа, будет $\frac{1}{1000 \text{ ГЦ}} \cdot 100 = 0,1\,c$.

Программа на данном этапе будет запускаться, показывать на мгновение окно с графическим интерфейсом, отрабатывать набор заданных в окне Block Diagram команд и сразу закрываться. Чтобы программа работала постоянно, необходимо, чтобы некоторый набор команд находился в бесконечном цикле. Для этого в *Block Diagram* необходимо выбрать

Functions/PROGRAMMING/Structures/While Loop и выделить все элементы в рамку цикла. Для остановки цикла необходимо в *Front Panel* установить кнопку остановки работы программы (*Controls/Express/ Stop Button).* В окне *Block Diagram* появившуюся иконку *Stop Button* соединить с красным кружком-терминалом, который находится в правом нижнем углу. Должна получиться программа, схожая с изображенной на рис.1.31.

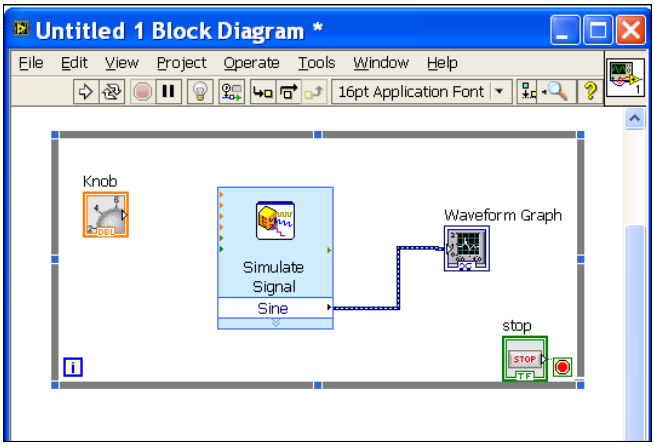

Рис.1.31. Создание бесконечного цикла в программе.

Аналогичный эффект достигается перетаскиванием на блок-схему компонента *Express/Exec Control/ While Loop* – в нём уже имеется кнопка *Stop*, подсоединённая к красному кружку-терминалу, обеспечивающему условие остановки цикла.

Для запуска проекта необходимо нажать на иконку в виде стрелки на верхней панели (либо пункт меню *Operate/Run*). После проделанных операций на осциллографе должен получиться сигнал, форма и амплитуда которого задана генератором (рис.1.32). Для остановки программы необходимо нажать кнопку *Stop*. Во время работы программы изображение будет “бежать” по экрану осциллографа, т.к. частота генерации сигнала 10,1 Гц, т.е. значение периода 0.099 с, а промежуток времени на экране 0,1 с. Если установить частоту генерации 10 Гц, “дрожание” сигнала на экране исчезнет.

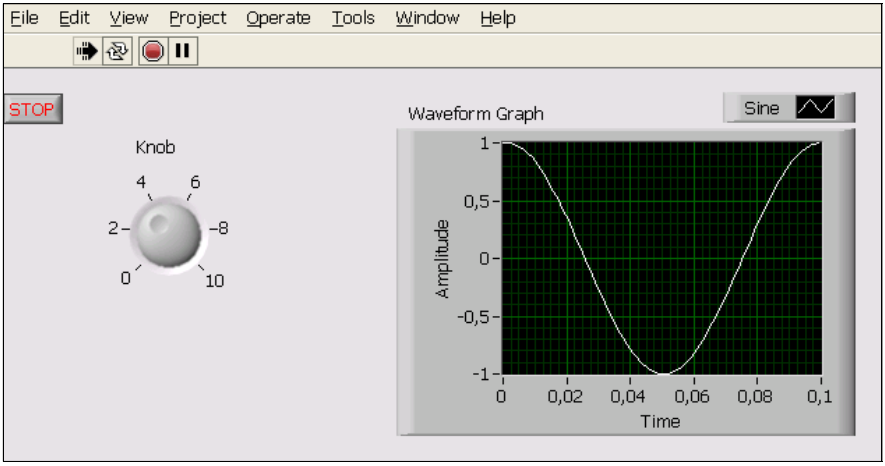

Рис.1.32. Запущенная программа осциллографа, подключенного к генератору.

Для регулировки амплитуды сигнала, поступающего с виртуального генератора, растянем мышкой на блок-схеме изображение генератора, чтобы были видны все его входы, подсоединим выход ручки *Knob* к входу *Amplitude* генератора (рис.1.33).

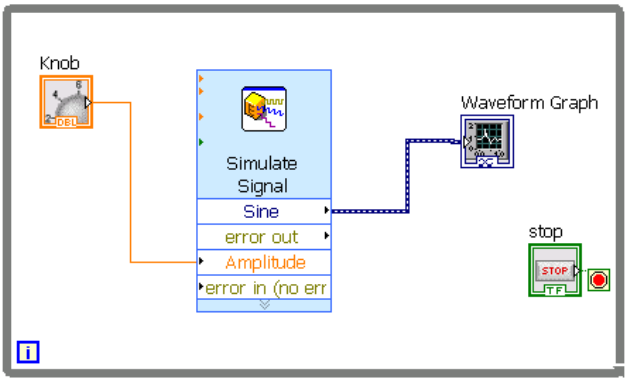

Рис.1.33. Регулировка амплитуды на выходе виртуального генератора.

Обратите внимание на то, что провод, по которому идёт сигнал, зависящий от времени (динамические данные), показывается более толстой линией с пунктиром внутри, а провод с выхода ручки *Knob* с не зависящим от времени

сигналом показывается тонкой сплошной линией (рис.1.33). Таким образом, на блок-схеме легко прослеживать ход сигналов разного типа.

При запуске разработанной программы поворот ручки будет задавать амплитуду генерируемого сигнала.

3. Порядок выполнения работы

Задание 3.1. Виртуальный калькулятор

Воспроизведите действия, описанные в теоретической части в разделе 2.1.

Сохраните проект в файл с именем “Лабораторная №1 Задание 3_1.vi”.

Задание 3.2. Виртуальные генератор и осциллограф - показ сигнала с виртуального генератора

Воспроизведите действия, описанные в теоретической части в разделе 2.2.

Сохраните проект в файл с именем “Лабораторная №1 Задание 3_2.vi”.

Задание 3.3. Виртуальные генератор и осциллограф - регулировка частоты и формы входного сигнала

Создайте копию предыдущего проекта и сохраните ее в файл с именем “Лабораторная №1 Задание 3_3.vi”. Для этого выберите пункт меню File/Save As…– в появившемся окне с вопросами ничего не меняйте, нажмите кнопку Continue… (продолжить), после чего в появившемся окне выбора файла введите новое имя *Лабораторная №1 Задание 3_3.vi* и нажмите кнопку OK.

Измените в свойствах генератора параметры сигнала (частоту и форму), и посмотрите, как изменяется сигнал на осциллографе.

Добавьте компонент *Express/Num Ctrls/Num Ctrl*, с выхода которого подайте сигнал на вход *Frequency* частоту генерируемого сигнала. Значение частоты задаётся в виде числа в поле ввода компонента *Num Ctrl* (на блок-схеме он имеет надпись *Numeric*). Для завершения ввода в поле этого компонента во время работы программы надо нажать на клавиатуре клавишу “Ввод” (Enter).

Сохраните проект.

***Замечание**:* значение, выдаваемое с ручки *Knob*, или с ползункового регулятора, или с других компонентов *Numeric Controls*, рассматривается как

сигнал, величина которого регулируется поворотом ручки, передвижением движка, и т.п.

Задание 3.4. Установка масштабов шкал и индикация величины сигнала

Создайте копию предыдущего проекта и сохраните ее в файл с именем “Лабораторная №1 Задание 3_4.vi”. Добавьте ещё один компонент *Knob* (ручка). Зайдите в его свойства (*Properties*) во вкладку Scale (шкала) и измените имеющуюся шкалу 0..10 ручки на -1..1. Добавьте индикатор *Express/Num Inds/ Thermometer* и измените в нём шкалу с 0..100 на -1..1. Замените надпись "*Thermometer*" на "Индикатор".

Подсоедините к индикатору выход *Knob*. Запустите приложение и пронаблюдайте, что при вращении ручки *Knob* изменяются показания индикатора.

Аналогичным образом можно изменять шкалы всех числовых компонентов.

Сохраните проект.

Задание 3.5. Виртуальный осциллограф - регулировка амплитуды входного сигнала

Создайте копию предыдущего проекта и сохраните ее в файл с именем “Лабораторная №1 Задание 3_5.vi”.

Величину сигнала, поступающего с выхода генератора, необходимо в последующей цепи умножить на значение, задаваемое ручкой дополнительного компонента *Knob* (см. предыдущее задание), и полученный сигнал вывести на экран осциллографа.

При выполнении задания необходимо использовать удаление и разветвление проводов (см. пункт “Работа с виртуальными проводами” в разделе 1.4.1).

Запустите программу на выполнение. Проверьте, происходит ли регулировка сигнала по амплитуде при вращении ручки. Сохраните проект.

Задание 3.6. Виртуальное устройство, которое показывает, сколько осталось времени до конца занятий (в часах и минутах)

а) Создайте виртуальный прибор, в котором имеется два пункта ввода *Numeric* для задания с клавиатуры текущего времени (в часах и минутах) и два пункта ввода для задания с клавиатуры времени окончания занятия (в часах и минутах). На индикатор должно выводиться число минут, которое осталось до конца занятия.

***Рекомендация**:* воспользуйтесь компонентами из разделов *Numeric Controls* и *Arith&Comparison*.

Замечание: в разделе *Functions/Programming/Timing/* имеются специальные компоненты, позволяющие работать с датой и временем. Однако на данном этапе освоения LabView целесообразно освоить работу с более простыми компонентами.

б) Усовершенствуйте предыдущий пример, обеспечив невозможность ввода отрицательных значений в пункты ввода, а также числа часов более 24 и числа минут более 60.

***Подсказка**:* воспользуйтесь меню *Properties*, вкладка *Data Entry* (“ввод данных”). Снимите флажок *Use Default Limits* (“Использовать границы значений по умолчанию”) и установите в поле задания минимального значения 0.

в) Усовершенствуйте предыдущий пример, добавив индикатор, на который выводится целая часть от числа часов, оставшихся до конца занятия.

***Подсказка**:* для округления воспользуйтесь компонентом *Round To -Infinity* (“округлять в сторону минус бесконечности”, т.е. для положительных чисел отбрасывать дробную часть числа).

г)* Усовершенствуйте предыдущий пример, добавив индикатор, на который выводится время в часах и минутах, оставшееся до конца занятия.

***Подсказка**:* для вычисления числа минут используйте числовые значения, полученные в частях а) и в) данного задания.

Сохраните проект.

Задание 3.7. Виртуальные часы

Создайте виртуальный прибор, в котором показывается текущее время. Для измерения времени используйте компонент *Functions/Programming/ Timing/Get Date/Time String*. Для вывода времени используйте компонент *String Indicator*. Для того, чтобы при индикации показывались секунды, ко входу *? want seconds* (“? нужны секунды”) подключите булевскую константу *True*. Сохраните проект.

Задание 3.8. Расчет электрической цепи с потенциометром

а) Нарисуйте в рабочем журнале электрическую цепь с подачей напряжения с источника напряжения на потенциометр и измерением напряжения между ползунком потенциометра и “землей”.

Напишите формулу зависимости выходного напряжения на ползунке потенциометра от положения ползунка (сопротивления между ползунком и заземленным выводом потенциометра).

В соответствии с этой формулой создайте виртуальное устройство, демонстрирующее, как изменяется выходное напряжение потенциометра в зависимости от положения ползунка. Подайте это виртуальное напряжение на виртуальный осциллограф *Waveform Chart* (с компонентом *Waveform Graph* схема работать не будет, так как на него можно подавать только массив – например, генерируемый в цикле). Подберите масштаб оси Y осциллографа так, чтобы было видно изменение амплитуды сигнала при перемещении ползунка (не забудьте убрать галочку “*AutoScale*” – автоматический выбор масштаба).

Воспользуйтесь обозначениями

E – переменная, соответствующая Э.Д.С. источника напряжения. Значение должно регулироваться в пределах от 0 до 10 В.

$\mathbf{U_{вых}}$ - выходное напряжение потенциометра.

$\mathbf{R_{max}}$ - полное сопротивление потенциометра. Значение должно задаваться в соответствии с указанием преподавателя (может лежать в пределах от 10 Ом до 10 Ком.

R - сопротивление между ползунком и заземленным выводом потенциометра.

***Замечание*:** обратите внимание, что ни в коем случае нельзя говорить про “виртуальный потенциометр”, или “виртуальный конденсатор”, или “виртуальный транзистор”, или “виртуальную катушку индуктивности”. Виртуальные приборы в среде LabView являются аналогами микросхем с выводами, являющимися либо только входами, либо только выходами. Например, если считать выводы резистора входами (так как на них можно подавать входные сигналы), то у него нет выходов, на которых появляются выходные сигналы, величина которых зависит от входных. А смешивать вход с выходом нельзя. Поэтому правильно говорить про виртуальный прибор, осуществляющий расчет токов и напряжений в электрической цепи, а элементы схемы должны соответствовать численному алгоритму расчета.

б*) Усложните предыдущее задание: учтите внутреннее сопротивление вольтметра, измеряющего значение $\mathbf{U_{вых}}$. В каких пределах нужно задать внутреннее сопротивление вольтметра, чтобы измерять значения $\mathbf{U_{вых}}$ с заданной точностью (лежащей в диапазоне от 0,1% до 1%)? Продемонстрируйте преподавателю полученные результаты с помощью вашего виртуального устройства.

Сохраните проект.

Задание 3.9. Использование Case Structure

Воспроизведите виртуальное устройство, схема которого показана на рис.1.17. Выходные данные с *Case Structure* подайте на индикатор *Meter.* Не забудьте задать разумное нижнее значение шкалы индикации. Сохраните проект.

Задание 3.10. Извлечение квадратного корня из вводимого числа

Создайте виртуальное устройство, в котором рассчитывается значение квадратного корня из значения, устанавливаемого с помощью компонента *Dial* (“циферблат”). Если число отрицательное, должно выводиться сообщение об ошибке. Для принятия решения о том, выводить значение числа или сообщение об ошибке, использовать *Case Structure*.

Сохраните проект.

***Подсказка**:* для вывода использовать компонент *String Indicator*. Для проверки знака входных данных использовать компонент *Greater Or Equal?* Для преобразования числа в строку использовать *Number To Fractional String.*

II. Лабораторная работа № 2-NI. Программирование виртуальных приборов в среде LabView

Лабораторная работа предназначена для развития и углубления навыков создания виртуального прибора в среде LabView, полученных студентами при выполнении лабораторной работы №1-NI. При ее выполнении студенты на примере решения конкретных задач получат дополнительные сведения о программировании в среде LabView, в том числе об использовании компонентов, находящихся в разделе Functions/Programming.

1.Теоретическая часть

1.1 . Некоторые элементы программирования LabView, использующиеся в работе

1.1.1. Массивы

При работе с группами данных и при накоплении данных после повторяющихся вычислений удобно использовать массивы. Массивом *(*Array*)* в LabView называется совокупность элементов одного типа. Элементами массива могут быть данные любого типа (например, строковые элементы, числа, элементы Boolean, массивы, кластеры, элементы управления). Не следует путать массив с элементами управления *Array* подраздела *Array, Matrix & Cluster/Array* разделов *Controls/Modern, Controls/Silver u Controls/Classic.* Эти элементы управления предназначены для показа на лицевой панели содержимого массивов.

Все элементы массива упорядочены. Обращение к отдельному элементу массива осуществляется через индекс, который находится в диапазоне от 0 до N-1, где N - число элементов в массиве. Массив может быть одномерным, двумерным и многомерным (размерности 3 и более). Чаще всего используются одномерные и двумерные массивы.

На блок-схеме передача массива с выхода одного компонента на вход другого изображается утолщенным проводником, число линий в котором зависит от размерности массива.

Провод, передающий одномерный массив, состоит из одной линии. Линия может иметь форму извилистой линии или пересечения извилистых линий, если передаются дискретные данные. К ним, например, относятся булевские и строковые данные.

Провод, передающий одномерный массив, изображается одной линией, двумерный – состоит из двух линий, трехмерный – тоже из двух линий, но находящихся на большем расстоянии друг от друга, и т.д. Цвет провода соответствует типу элементов массива.

Провода для передачи целочисленных данных имеют светло-синий цвет:

– передается целая величина;

– передается одномерный целочисленный массив;

– передается двумерный целочисленный массив.

Провода для передачи булевских данных имеют зеленый цвет:

– передается булевская величина;

– передается одномерный булевский массив;

– передается двумерный булевский массив

Провода для передачи строковых данных имеют малиновый цвет:

– передается строковая величина;

– передается одномерный массив строк;

– передается двумерный массив строк.

На провода для передачи массивов очень похожи провода, через которые передаются динамические данные или кластеры данных, но их не следует путать:

– передаются динамические данные, цвет провода темно-синий;

– передается кластер данных, цвет провода коричневый.

Динамические данные – сигналы, возникающие через заданные промежутки времени (например, с выхода генератора Simulate Signal).

Кластеры данных в LabView являются аналогами записей языка PASCAL или структур языка C и предназначены для группировки в единое целое данных разных типов.

В отличие от проводов для передачи массивов провода для передачи динамических типов данных имеют темно-синий цвет и внутри заполнены пунктиром из белых точек, а для кластеров данных – коричневый цвет и внутри также заполнены пунктиром из белых точек.

Все элементы массива различаются только значениями, все свойства элементов массива (размер, точность, представление, цвет проводника на блок-схеме) одинаковы. Если изменить свойства у одного из элементов массива, то аналогично изменяются свойства у всех остальных элементов.

Для работы с массивами в LabView служат различные функции, которые находятся на панели *Functions/Programming/Array/* (Рис.2.1).

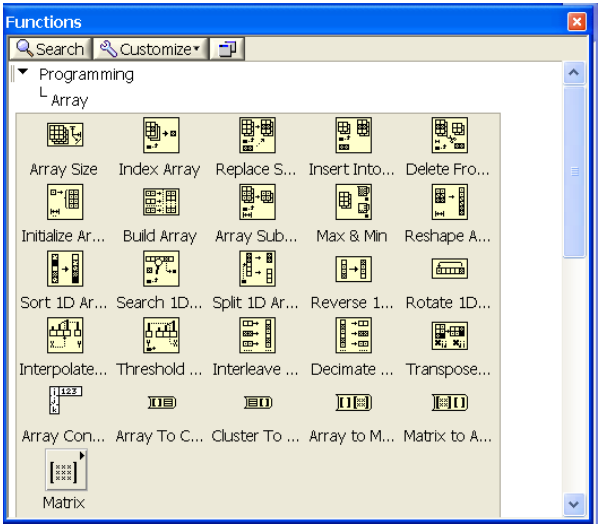

Рис.2.1. Функции раздела *Functions/Programming/Array/.*

Многие компоненты LabView используют массивы. Например, на вход компонента Table для показа текстовой таблицы подается двумерный массив строк.

Поскольку тип данных выхода одного компонента должен совпадать с допустимым типом данных для входа другого, необходимо, чтобы массивы имели одинаковую размерность и содержали элементы одного типа. Для

согласования типов требуется использовать компоненты преобразования типа.

При использовании массивов для преобразования элементов массива в число и наоборот можно воспользоваться двумя взаимно-обратными функциями:  - *Number to Boolean Array* и  - *Boolean Array to Number,* расположенными во вкладке *Functions/Programming/Numeric/ Conversion/*. Функция *Number to Boolean Array* преобразует целое число в двоичный код, состоящий из ноликов и единиц, рассматриваемый как массив логических переменных. Функция *Boolean Array to Number* осуществляет обратное преобразование.

1.1.2. Компонент Formula

Компонент *Express/Arithmetic&Comparison/Formula* используется для преобразования двух или более сигналов по заданной формуле, в которую входят эти сигналы.

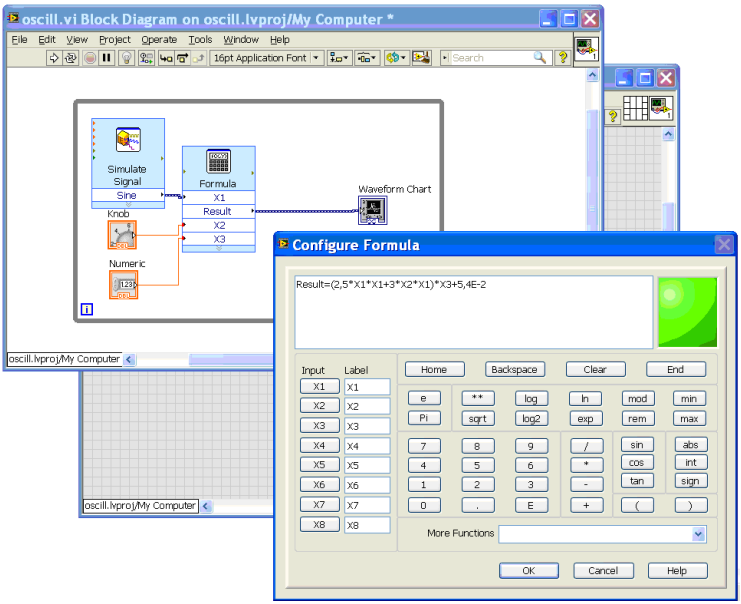

Рис.2.2. Использование компонента *Formula* для сложной математической обработки сигналов от нескольких источников.

В поле ввода формулы (рис.2.2) задаётся выражение, в котором переменной *Result* (ей соответствует выход *Result*) присваивается выражение, зависящее от переменных X1,X2,…,X8, соответствующих сигналам на входах X1,X2,…,X8.

Использование компонента *Formula* заметно упрощает вычисление сложных математических выражений.

1.1.3. Структура Formula Node

С помощью узла *Formula Node* (*Functions/ Programming*/Structures/ *Formula Node* – рис.2.3) можно не только ввести в окно компонента одну или несколько формул, но и написать программный код с использованием операторов языка С – например, осуществить выбор данных по условию внутри узла. Однако, в отличие от компонента *Formula*, компонент *Node Formula* не может использоваться для обработки динамического потока данных – например, с генератора сигналов *Simulate Signal.*

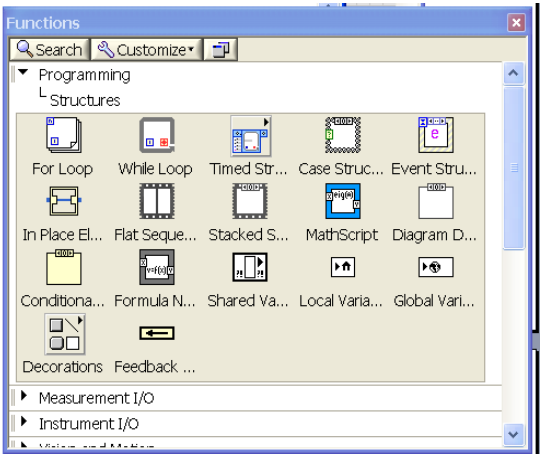

Рис.2.3. Вид раздела *Functions/Programming/Structures* с узлом *Formula Node.*

Рамку узла необходимо растянуть до нужного размера и в нее вписываются формулы и программный код (рис.2.4).

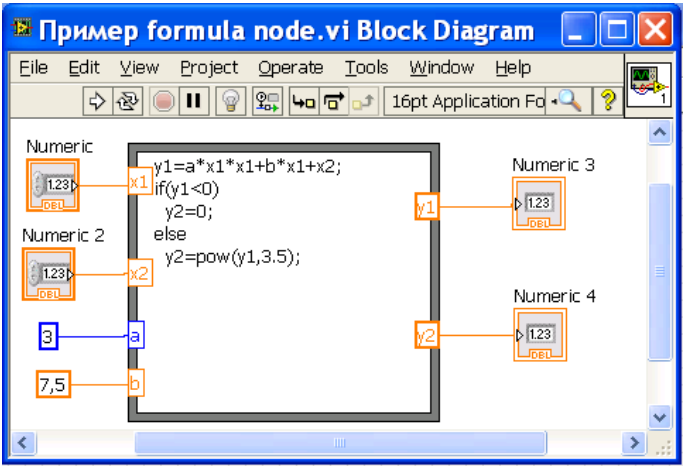

Рис.2.4 Пример вычислений с помощью узла *Formula Node.*

Можно использовать встроенные функции (см. в Help раздел "*Formula Node* and *Expression Node Functions*" – различные варианты округления, abs, exp, прямые и обратные тригонометрические, прямые и обратные гиперболические, логарифмические). Для извлечения квадратного корня из x можно использовать функцию sqrt(x), для возведения x в степень y - функцию pow(x,y), и т.д.

Необходимо обратить внимание на то, что при записи значений чисел в тексте программы в качестве десятичного разделителя всегда используется точка, хотя в арифметических константах используется национальный разделитель, в нашем случае запятая (рис.2.4).

Входные и выходные терминалы узла *Formula Node* можно создать, щелкнув правой кнопкой мыши по границе узла и выбрав опцию *Add Input* (Добавить ввод) или *Add Output* (Добавить вывод) в контекстном меню. Эти терминалы являются аналогами переменных в списке параметров подпрограмм, написанных на процедурных языках программирования (PASCAL, C и т.п.), а сама структура *Formula Node* является аналогом подпрограммы.

Для ускорения процесса создания алгоритма можно копировать и вставлять имеющиеся фрагменты текстов программ на языке C.

Неизвестные пишутся в левой части формул. При наличии нескольких формул каждая формула пишется с новой строки и заканчивается точкой с запятой.

На рис.2.4 показан пример кода для вычисления значений y1 и y2.

1.1.4. Особенности программирования виртуальных приборов, связанные с наличием частоты дискретизации

В отличие от аналоговых устройств, в которых значения величин меняются со временем непрерывно, цифровые устройства проводят изменение или измерение сигналов дискретно, с некоторой частотой, называемой *частотой дискретизации*.

Цифровой генератор *Simulate Signal* проводит установку новых значений на выходе с частотой дискретизации, задаваемой в его свойстве *Samplers per second* (Hz) – точек в секунду (Гц).

Цифровые осциллографы на основе компонентов *Waveform Graph* и *XY Graph* показывают точки графика с частотой дискретизации, соответствующей входному сигналу. У компонента *Waveform Graph* только один вход, поэтому на него сигналы с разной частотой дискретизации подать невозможно. А вот в случае подачи на входы компонента *XY Graph* двух сигналов с разной частотой дискретизации LabView выдает сообщение об ошибке, так как в этом случае невозможно построить график Y(X), т.е. Y[i] от X[i] (где i – номер точки). Ведь точки Y[i] и X[i] будут соответствовать разным моментам времени.

1.1.5. Отладка ошибок

При работе над проектами, особенно если они сложные, неизбежны ошибки. Проверка ошибок позволяет узнать, в каком месте и по какой причине произошел сбой в работе виртуального прибора.

Ошибки могут быть как достаточно сложными, требующими для их исправления специальных средств LabView, так и совсем простыми. Наиболее распространенные ошибки бывают двух типов, которые мы назовем условно синтаксическими и логическими.

При наличии *синтаксической ошибки* кнопка запуска приобретает вид сломанной стрелки, и виртуальный прибор не запускается. Ошибка показывается во вкладке *Error List* (Список ошибок) и высвечивается на блок-схеме. К числу синтаксических ошибок относятся, например, такие:

- на блок-схеме у компонента не подключен вход, требующий обязательного ввода данных (например, у какого-либо арифметического или логического компонента – *Add, Subtract, And, Or* и т.п.);
- соединены выходы двух или более элементов управления;
- на блок-схеме есть неисправные провода (не подсоединенные, либо оборванные, либо с несоответствующим типом данных);
- подача на входы компонента, требующего синхронных сигналов (например, компонента *XY Graph*), двух сигналов с разной частотой дискретизации.

Логическая ошибка – когда разработчик написал программу, которая делает не то, или не совсем то, или не всегда то, что ожидалось.

При наличии логической ошибки виртуальный прибор запускается, но работает некорректно. Для исправления этого типа ошибок используется пошаговый режим отладки программы. Логическая ошибка возникает, например, если программа должна вычислить число точек измерения, а показывается значение 10,5. Не может быть “половины точки измерения”! Если данное значение использовать в дальнейшей работе программы для проведения измерений, могут получиться результаты, которые будут казаться правдоподобными, но неверными из-за допущенной логической ошибки.

Прежде чем обсуждать способы исправления ошибок, рассмотрим назначение некоторых кнопок инструментальной линейки (рис.2.5).

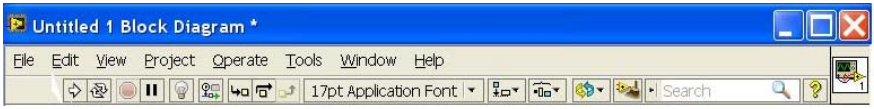

Рис.2.5. Вид инструментальная линейки блок-схемы.

 - кнопка **Run** (Запуск).

 - кнопка **Continuous Run** (Непрерывный запуск). При нажатии на эту кнопку программа выполняется непрерывно до нажатия на кнопку **Abort**, которая находится на лицевой панели.

 - кнопка **Abort** (Прервать). При нажатии на эту кнопку программа будет остановлена, но данные могут быть потеряны!

 - кнопка **Pause** (Пауза). Временно останавливает работу программы, после чего можно использовать одношаговые операции отладки программы, используя описанные ниже кнопки пошагового выполнения. Для того чтобы вновь запустить программу, нужно еще раз нажать эту кнопку.

 - кнопка **Step Into** (Шаг внутрь), войти внутрь узла (используется только в режиме отладки).

 - кнопка **Step Over** (Шаг через), шаг вперед.

 - кнопка **Step Out** (Шаг наружу), выйти изнутри узла.

 - кнопка **Execution Highlighting** (Подсветка выполнения). Подсвечивает поток данных, проходящих через блок-схему, что позволяет увидеть промежуточные величины данных, появляющихся по мере выполнения программы и визуально проследить за данными во время их перехода из одного узла в другой.

При попытке запустить неисправную или незаконченную программу кнопка запуска принимает вид сломанной стрелки. Чтобы найти и исправить ошибки, нужно щелкнуть по сломанной стрелке мышкой, появится окно *Error List* (Список ошибок), в котором содержится описание ошибок (рис.2.6).

Чтобы получить более подробное описание конкретной ошибки, нужно щелкнуть по соответствующему пункту списка мышкой (рис.2.6).

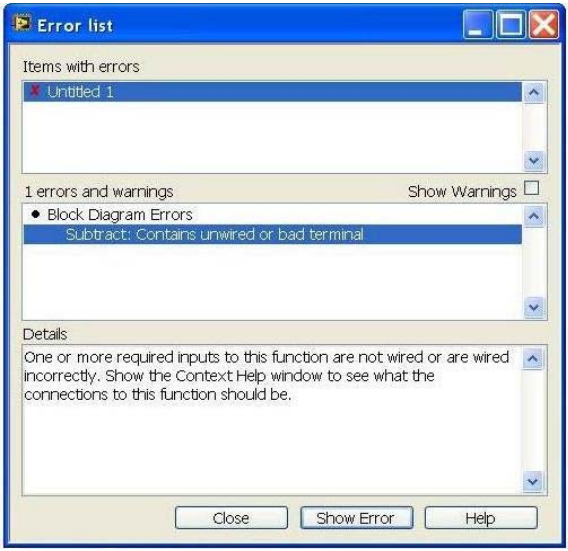

Рис.2.6. Вид окна со списком ошибок

Для определения местонахождения ошибки на блок-схеме нужно сделать двойной щелчок по ошибке в списке, либо зайти в раздел *Properties*. LabView выведет соответствующее окно на экран и выделит подсветкой объект, вызвавший данную ошибку. После исправления ошибки нужно снова запустить программу.

Во время отладки для того, чтобы быстрее дойти до нужного места на блок-схеме, полезно выполнять программу пошагово, реализуя работу узла за узлом. Чтобы начать пошаговое выполнение, следует запустить виртуальный прибор щелчком мыши по одной из кнопок пошагового выполнения (вместо кнопки Run**),** затем можно временно остановить виртуальный прибор, например, щелкнув мышью по кнопке Pause**.**

Каждая кнопка определяет способ выполнения следующего шага. При нажатии на кнопку **Step Into** выполняется заход внутрь узла (структуры или подпрограммы – виртуального подприбора, SubVI) и осуществляется первый шаг работы алгоритма выбранного узла.

При нажатии на кнопку **Step Over**, “шаг вперед”, происходит выполнение всего алгоритма узла, и на выходе узла появляются результаты.

При нажатии кнопки **Step Out** (Шаг наружу) происходит выполнение работы узла, появление на выходах узла результатов, выход из алгоритма узла и остановка перед началом выполнения алгоритма следующего узла.

В LabView можно наблюдать за прохождением данных в процессе выполнения или отладки программы, используя кнопку **Execution Highlighting** (Подсветка выполнения) – режим “Лампочка” на блок-схеме. Движение данных из одного узла в другой отмечается перемещающимися вдоль проводников кружочками. Во время работы в режиме подсветки на выходных терминалах автоматически отображаются исходящие данные.Это удобный и красивый способ анимации потоков данных, но программа в этом режиме выполняется гораздо медленнее. Выключается режим визуализации данных повторным нажатием на ту же кнопку.

2. Демонстрационные примеры

2.1. Демонстрационный пример 1 – создание виртуального преобразователя сигналов с использованием элемента Formula

Воспользуйтесь компонентом Formula для преобразования сигнала по заданной формуле. Создайте виртуальный прибор, содержащий виртуальные генератор и осциллограф. Предусмотрите возможность изменения амплитуды и частоты сигнала с генератора.

Замечание: воспользуйтесь компонентом *Waveform Chart*.

В данном задании необходимо выводить на экран осциллографа зависимость от времени величины, вычисляемой по формуле (2.5*X1*X1+3*X2*X1)*X3+0.054, где значение X1 – сигнал с генератора, а X2 и X3 – числовые параметры, задаваемые пользователем программы. Добавьте компоненты *Numeric* для установки значений X2 и X3.

Вызовите компонент *Formula* из раздела *Express/Arithmetic&Comparison* (рис.2.7).

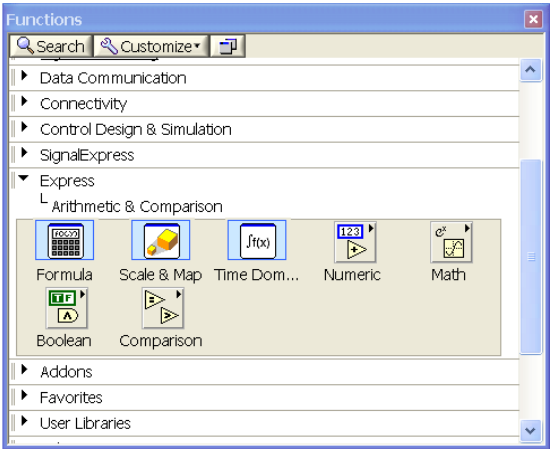

Рис.2.7. Вид раздела *Express/Arithmetic&Comparison* с компонентом *Formula*.

Щелкните дважды левой кнопкой мыши по значку *Formula* на компоненте, появится окно ввода, в которое нужно вписать необходимую формулу *Result*= (2.5*X1*X1+3*X2*X1)*X3+0.054.

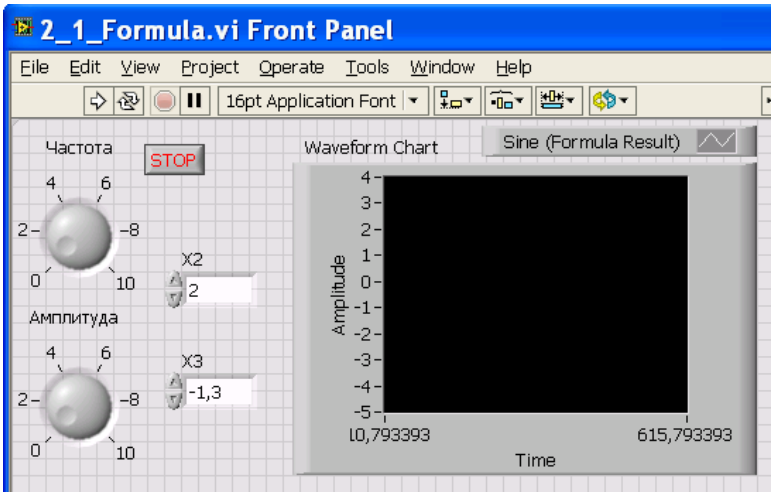

Рис.2.8. Вид лицевой панели виртуального преобразователя сигналов.

Вид лицевой панели получившегося виртуального прибора показан на рис.2.8, вид блок-схемы программы – на рис.2.9.

Переменные X1, X2 и X3 соответствуют сигналам на входах, переменной *Result* соответствует выход *Result* узла *Formula*, который соединяется с индикатором *Waveform Chart*.

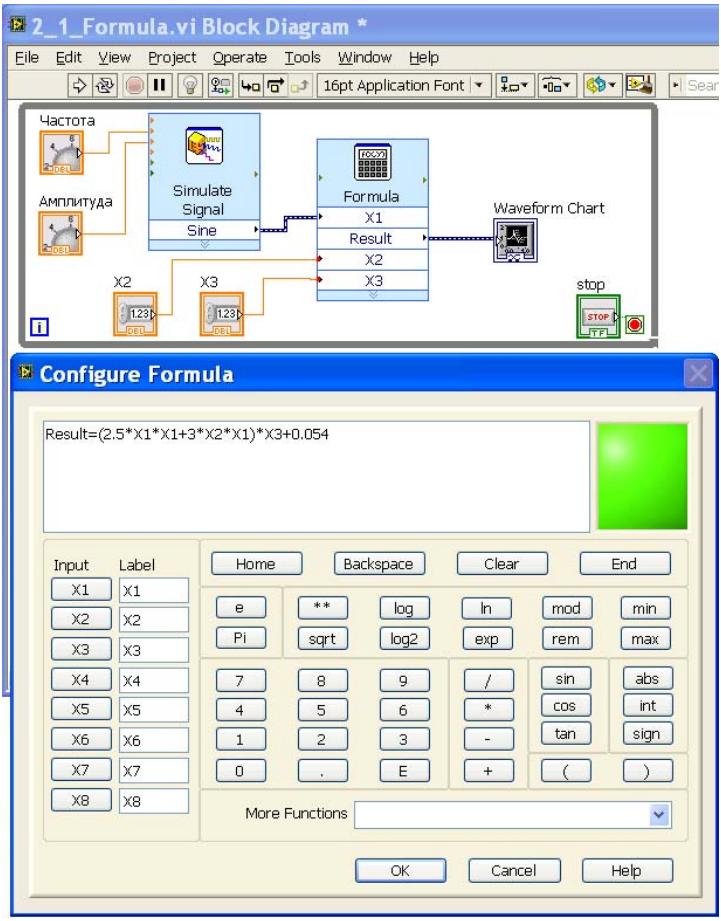

Рис.2.9. Использование компонента *Formula* для сложной математической обработки сигналов от нескольких источников.

Запустите программу, проследите за тем, как меняется выходной сигнал при изменении параметров входных сигналов.

2.2. Демонстрационный пример 2 - отладка программы с ошибками

Продемонстрируем отладку программы с ошибками. Создайте виртуальный прибор, который осуществляет следующий алгоритм:

случайное число, генерируемое компонентом *Functions/Programming/ Numeric/Random Number* (0-1), умножить на константу, прибавить к результату другую константу и вывести результат на *Waveform Chart*.

Компонент *Random Number* (0-1) изображается в виде двух игровых кубиков с точками на гранях, символизирующих выпадение случайных чисел. Он реализует алгоритм генерации случайного числа в диапазоне от 0 до 1 (включая 0 и не включая 1), т.е. генерирует вещественные числа, содержащие дробную часть. У компонента отсутствуют какие-либо входные параметры, есть только выходной – сгенерированное случайное число.

Сделайте вывод на индикаторы следующих данных: 1) случайного числа, 2) результата умножения этого числа на константу, 3) результата сложения этого результата со второй константой.

На блок-схеме сделайте ошибки: например, не подсоедините константу к элементу сложения, а также добавьте на схему неиспользуемый логический элемент And.

Убедитесь в том, что стрелка кнопки **Run** приобрела сломанный вид. Проделайте следующие действия:

1.Щелкните по сломанной стрелке, найдите ошибку в списке. Выделите место ошибки, нажав на кнопку **Show Error** внизу списка**,** подсоедините константу, запустите программу, убедитесь, что после этого вместо двух ошибок показывается одна. Устраните и эту ошибку и убедитесь, что виртуальный прибор работает.

2.Отсоедините индикатор результата сложения. Запустите программу, убедитесь в том, что она работает, стрелка **Run** имеет нормальный вид, но результат сложения не выводится на индикатор – программа работает некорректно.

3.Расположите лицевую панель и блок-схему виртуального прибора так, чтобы они не перекрывали друг друга. Запустите программу и пронаблюдайте изменение значений на индикаторах. Включите подсветку выполнения (кнопка **Execution Highlighting**) и пронаблюдайте

последовательность прохождения данных от узла к узлу, а также изменение значений на индикаторах.

4. Проверьте работу узлов программы в пошаговом режиме: запустите программу и приостановите выполнение нажатием кнопки **Pause**. После чего продолжите выполнение программы, нажимая соответствующие кнопки **Step Into**, **Step Over, Step Out** на линейке инструментов блок-схемы. Нажимая несколько раз кнопку **Step Over**, проследите, как последовательно выполняются алгоритмы узлов программы. Обратите внимание на анимацию продвижения данных от одного узла к другому (при включенном режиме подстветки), и на появление данных на лицевой панели по мере пошагового выполнения программы. При поступлении новых данных на терминалы блок-схемы, соответствующие индикаторам, обновляется содержимое индикаторов на лицевой панели.

Также полезно обратить внимание на то, что подсказки-названия кнопок **Step Into, Step Over, Step Out** на линейке инструментов блок-схемы во время выполнения меняются. Если очередной узел представляет один оператор, т.е. не является структурой или подпрограммой, подсказка для кнопки **Step Into** будет "**Step Over**" и далее название следующего узла, как и для кнопки **Step Over**. Этому не следует удивляться и считать, что что-то работает неправильно.

3. Компьютерный тест №2

В тесте проверяется усвоение учащимся теоретической части описания лабораторной работы. Для выполнения теста следует зайти на сайт http://distolymp2.spbu.ru/www/olymp/ под своим логином и перейти по ссылке "Компьютерный тест "№2".

После прохождения теста в разделе «Результаты» правильные ответы помечаются зеленым цветом, неправильные – красным. Результаты прохождения теста служат допуском к выполнению лабораторной работы.

4. Порядок выполнения работы

Перед выполнением заданий воспроизведите действия, описанные в разделах 2.1 и 2.2 «Демонстрационные примеры».

При выполнении работы заносите в рабочий журнал и в файл, который прикладывается к отчету, ответы на вопросы во всех заданиях, объяснение результатов, необходимые вычисления и комментарии.

Задание 4.1. Виртуальный смеситель сигналов - показ результата на экране виртуального осциллографа

Создайте три генератора с различными параметрами выходных сигналов. Обеспечьте вычисление произведение первого и второго сигнала и разность второго и третьего. Полученные сигналы подайте с полученных выходов на входы двух виртуальных осциллографов. Оформите лицевую панель виртуального прибора, присвойте имя каждому индикатору.

Сохраните проект в файле с именем “Лабораторная №2 Задание 4_1.vi”.

***Замечание**:* компонент, находящийся на блок-схеме, можно скопировать. Выделите виртуальный генератор, и для занесения в буфер обмена либо нажмите комбинацию клавиш Ctrl+C, либо воспользуйтесь пунктом меню *Edit/Copy*. Вставить компонент из буфера обмена можно комбинацией клавиш Ctrl+V, либо пунктом меню *Edit/Paste*.

Задание 4 .2. Демонстрация на экране виртуального осциллографа фигур Лиссажу

Фигура Лиссажу – замкнутая траектория точки, совершающей гармонические колебания в двух взаимно перпендикулярных направлениях. Если периоды относятся как целые числа, то через промежуток времени, равный наименьшему кратному обоих периодов, движущаяся точка снова возвращается в то же положение, и траектория является фигурой Лиссажу.

Вид фигур Лиссажу зависит от соотношения между частотами и фазами складывающихся колебаний. Наиболее красивые (симметричные) фигуры обычно появляются при различии в фазах колебаний 0, 90 или 180 градусов и при отношении частот равном отношению небольших целых чисел (1 и 1; 1 и 2; 2 и 3; 1 и 4; 3 и 4; и т.д.).

4.2.1. Фигуры Лиссажу для сигналов одной частоты

Создайте два генератора *Simulate Signal* с одинаковой частотой выходных сигналов (можно использовать значения по умолчанию).

Затем поместите на лицевую панель прибор для одновременного наблюдения двух синусоидальных функций – виртуальный двухлучевой осциллограф. Для этого удобно использовать компонент *XY Graph (Control/Express/Graph Indicator/XY Graph).*

Установите у компонента в качестве подписи осей имена X и Y.

Запустите программу и объясните результат.

Сохраните проект.

4.2.2. Фигуры Лиссажу – изменение фазы сигналов во время работы программы

Создайте копию предыдущего проекта – сохраните ее с именем “Лабораторная №2 Задание 4_2_2.vi”.

Зайдите в свойства первого генератора и установите для него сдвиг по фазе выходного сигнала 45 градусов.

Запустите программу и объясните результат.

Каждый раз останавливать программу для задания сдвига фаз генераторов неудобно. Поэтому добавьте на лицевую панель два компонента *Numeric* и подсоедините выход первого из них к входу *Phase* первого генератора *Simulate Signal*, а выход второго – к входу *Phase* второго генератора. Не забудьте предварительно растянуть на блок-схеме иконку генератора так, чтобы стали видны дополнительные входы, среди которых находится и вход *Phase*.

У компонента *Simulate Signal* имеется особенность – задание начальной фазы сигнала производится либо в начале работы программы, либо по сигналу *Reset* (). Поэтому изменение значений чисел в компоненте *Numeric* не будет влиять на фазу сигнала соответствующего генератора до тех пор, пока на вход *Reset* генератора не поступит логическое значение *true*. Проще всего его получать нажатием кнопки

Поместите кнопку *OK Button* на лицевую панель и соедините выход этой кнопки с входами *Reset* генераторов.

Запустите программу. Установите для первого генератора сдвиг по фазе выходного сигнала 45 градусов, а у второго генератора 0. Нажмите кнопку *OK Button* и объясните результат.

Установите для второго генератора сдвиг по фазе выходного сигнала 45 градусов. Нажмите кнопку *OK Button* и объясните результат.

Установите для второго генератора сдвиг по фазе выходного сигнала -45 градусов. Нажмите кнопку *OK Button* и объясните результат.

Сохраните проект.

4.2.3. Фигуры Лиссажу для сигналов разной частоты

Создайте копию предыдущего проекта – сохраните ее с именем "Лабораторная №2 Задание 4_2_3.vi".

Добавьте еще два компонента *Numeric* для задания частот генераторов, введите в каждом из них в качестве начального значения 10, т.е. частоту генерации 10 Гц.

Установите у компонента XY Graph в качестве подписи осей имена X и Y. Задайте для показа по осям X и Y фиксированные масштабы *Minimum=-1, Maximum=1.*

Замечание: если оставить автоматический выбор масштаба, при каждом выводе будут выбираться разные границы, и будет наблюдаться мерцающая картинка, не имеющая физического смысла. В предыдущих заданиях просто повезло, что во время каждого построения графика границы графика не менялись, и выводимые значения всегда лежали в диапазоне от -1 до 1. Для корректного показа картинки на виртуальном осциллографе по возможности следует выбирать фиксированные масштабы шкал.

Подберите частоты и фазы сигналов генераторов так, чтобы продемонстрировать возникновение фигур Лиссажу. Не забывайте нажимать на кнопку *OK Button* после изменения значений фазы.

Что произойдет, если фаза сигнала первого генератора 0, а второго 90 градусов?

Что произойдет, если частоту генерации увеличить до 100 Гц? Почему?

Что произойдет, если при частоте генерируемых сигналов 100 Гц в генераторах увеличить частоту дискретизации (*Samples per second* – точек в секунду) со значения по умолчанию (1000) до 10000? Почему?

Сохраните проект.

4.2.4. Фигуры Лиссажу для сигналов разной частоты – задержка в выполнении цикла

Создайте копию предыдущего проекта – сохраните ее с именем “Лабораторная №2 Задание 4_2_4.vi”.

Добавьте на блок-схему внутрь рамки основного цикла программы компонент *Time Delay* (задержка по времени в секундах), находящийся в разделе *Timing*, и в появившемся окне оставьте значение по умолчанию, т.е. 1 секунду.

После запуска программы перед каждым выводом порции данных на экран будет происходить задержка на 1 секунду, что позволит увидеть порядок смены кадров на цифровом осциллографе. Измените частоту одного из генераторов на такое значение, при котором раньше наблюдалось только мелькание. Посмотрите, как в режиме с задержкой будет рисоваться кривая на экране.

Сохраните проект.

4.2.5. Фигуры Лиссажу для сигналов разной частоты – автоматическая подача сигнала Reset на виртуальные генераторы

Создайте копию предыдущего проекта – сохраните ее с именем “Лабораторная №2 Задание 4_2_5.vi”.

Удалите со схемы компонент *OK Button.* Вместо сигнала с кнопки OK Button подайте на входы Reset виртуальных генераторов сигнал *True Constant* из паллеты *Boolean*. В результате после каждого срабатывания таймера будет появляться значение *True* на входе *Reset*, и будет устанавливаться новое значение начальной фазы генераторов, соответствующее новым цифровым значениям на их входах.

Наблюдать изменение фазы с помощью компонента *Numeric* может быть неудобно, так как он не обеспечивает плавной прокрутки в необходимом диапазоне значений. Для того чтобы плавно менять фазу в необходимом диапазоне значений, можно перейти от компонента *Numeric* к компоненту *Pointer Slide*. В LabView имеется очень удобная возможность замены одного элемента управления (*Control)* на другой без обрыва виртуальных проводов, подключенных к элементу управления.

Перейдите в окно лицевой панели, щелкните правой кнопкой мыши по компоненту *Numeric*, задающему фазу сигнала первого генератора, и выберите пункт меню *Replace*. В появившемся окне зайдите в раздел *Num Ctrls*, и выберите компонент *Pointer Slide* (ползунок с указателем) – и компонент *Numeric* будет заменен компонентом *Pointer Slide* как на лицевой панели, так и на блок-схеме.

Задайте параметры ползунка так, чтобы фазу можно было менять в пределах от 0 до 360. Проверьте работу программы, демонстрируя, как меняются фигуры Лиссажу при перемещении движка ползунка, задающего фазу.

Сохраните проект.

Задание 4 .3. Демонстрация на экране виртуального осциллографа результата сложения двух гармонических сигналов

4.3.1. Сложение гармонических колебаний одной частоты, но с разной амплитудой и фазой

Сделайте копию проекта 4_2_2.

Удалите компонент *XY Graph,* и в качестве цифрового осциллографа используйте компонент *Waveform Graph*. Подайте на его вход сумму сигналов с выходов двух генераторов гармонических сигналов, использовавшихся в прошлых заданиях.

Добавьте на лицевую панель еще два компонента *Pointer Slide* и используйте их для регулировки амплитуды сигналов на выходе генераторов.

Задайте частоту генерируемых первого и второго сигналов 50 Гц.

Задайте в первом и втором генераторах частоту дискретизации (*Samples per second*) равной 1000 точек в секунду (значению по умолчанию).

Задайте разумным образом максимальное значение шкал ползунковых регуляторов и фиксированное значение границ графика по оси Y, а по оси времени оставьте автоматическое выставление масштаба.

Запустите программу. Проверьте, какой формы сигнал получается при сложении двух гармонических сигналов одинаковой частоты. К чему приводит увеличение сдвига фазы первого генератора? Почему? Как повлияет на результирующий сигнал изменение амплитуды сигнала с первого генератора? Второго генератора?

Сохраните проект.

4.3.2. Проблема дискретности сигнала. Работа с графиками

Сделайте копию предыдущего проекта.

Увеличьте частоту сигналов с генераторов до 100 Гц. Форма сигнала на экране цифрового осциллографа станет не очень похожей на синусоиду, но выбранный автоматически масштаб по оси времени будет неудобным. Сменить масштаб можно без остановки программы. Для этого щелкните правой кнопкой мыши в любом месте компонента и уберите галочку напротив свойства *AutoScale X* (автоматическое масштабирование по оси X).

После этого можно редактировать минимальное и максимальное значения шкалы – прямо в области шкалы. При заданных условиях по оси времени это были 0 и 0,099. Минимальное значение менять не надо, а максимальное следует сменить на 0,02.

Замечание: режимы *AutoScale X* и *AutoScale Y* можно включать и выключать в любое удобное пользователю время. При включении режима автоматического выбора масштаба происходит растягивание графика на весь экран. Если после этого выключить данный режим, установленный масштаб сохранится. Так что часто бывает удобно включить режим *AutoScale* по нужной оси и сразу выключить.

После сделанного изменения при просмотре графика станет очевидно, что сигнал не является синусоидальным (рис.2.10). Однако причина этого исходя

из формы графика неочевидна – установлена форма показа графика по умолчанию, при этом точки графика соединены отрезками прямых, а сами точки не показываются.

Щелкните мышью в маленьком окошке, находящимся над правым верхним углом основного окна графика (около надписи *Sine*), и выберите в появившемся меню пункт *Point Style* (рис.2.10).

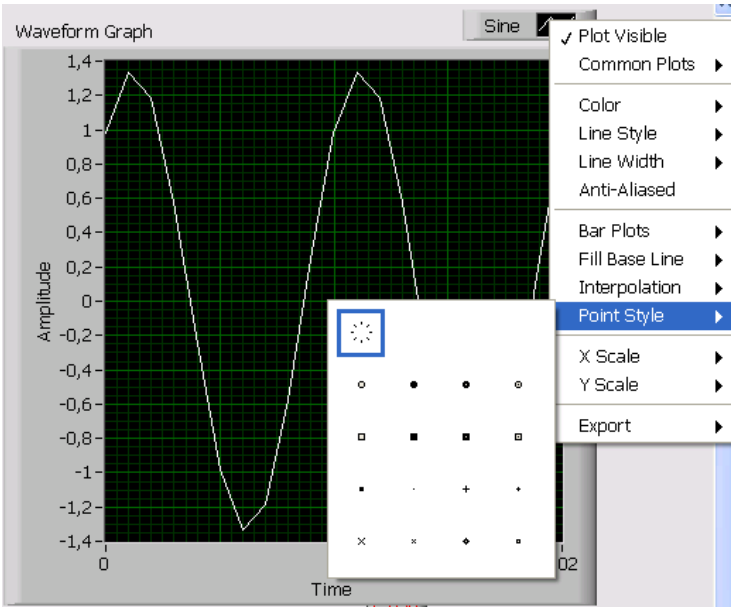

Рис.2.10. Задание параметров рисования графика.

По умолчанию выбран режим, в котором точки не показываются (символ, обведенный рамкой). Выберите любой вариант формы точки из первого или второго ряда, чтобы она была хорошо заметна. После этого точки сигнала, по которым строится график, будут хорошо видны. И становится понятным, что отличие формы сигнала от синусоидальной связано с дискретностью сигнала – точки идут не непрерывно, ставятся через равные промежутки времени, соответствующие *частоте дискретизации* генератора.

Установите нулевую амплитуду второго сигнала и единичную – у первого сигнала. Сдвиг фазы сигнала установите равным нулю.

Проверьте, как будет выглядеть график при частоте сигнала 250 Гц. Через какую часть периода сигнала ставится каждая точка? Какое количество точек генерируемого сигнала приходится на один период синусоиды? Будет ли меняться расстояние между точками по оси времени при изменении частоты или фазы сигнала?

Задайте нулевой сдвиг фазы сигнала. Почему при частоте сигнала 250 Гц форма сигнала кажется пилообразной? Если ответ на вопрос не ясен, нарисуйте в тетради график синусоиды, и поставьте на нем через каждые четверть периода точки, начиная с начала координат. А потом соедините соседние точки отрезками прямых.

Почему при частоте сигнала 250 Гц при изменении фазы сигнала на графике кажется, что амплитуда сигнала меняется? Если ответ на вопрос не ясен, нарисуйте в тетради график синусоиды, и поставьте на нем через каждые четверть периода точки, начиная с точки с некоторым сдвигом по фазе от начала координат. А потом соедините соседние точки отрезками прямых.

Задайте частоту сигнала с первого генератора 500 Гц и сдвиг фазы, равный нулю. Увеличьте сдвиг фазы. Объясните результат.

Задайте частоту сигнала с первого генератора 50 Гц и сдвиг фазы, равный нулю. Увеличьте сдвиг фазы. Объясните результат.

Что произойдет, если увеличить частоту выше 500 Гц?

Остановите программу и задайте первому генератору частоту дискретизации (*Samples per second*) 10000 точек в секунду. Попробуйте запустить программу. Что произойдет? Почему?

Задайте второму генератору также частоту дискретизации 10000 точек в секунду и запустите программу. Объясните результат.

До какой максимальной частоты сигнала можно проводить измерения при данной частоте дискретизации (10 КГц)?

Можно ли установить частоту дискретизации генераторов 1 ГГц (1E9)? Если да, то как будет выглядеть график сигнала с частотой 1E8 Гц?

Сохраните проект.

4.3.3. Сложение гармонических колебаний разной частоты. Биения

Сделайте копию предыдущего проекта.

Подберите параметры первого сигнала так, чтобы на экране было видно 30-40 периодов. Для такого режима показ точек сигнала лучше убрать.

Установите амплитуду второго сигнала примерно равной амплитуде первого и подберите частоту второго сигнала так, чтобы при сложении сигналов продемонстрировать биения.

Что произойдет, если менять фазу одного из сигналов? Уменьшать амплитуду одного из сигналов?

Если частоты складываемых сигналов близки, итоговый сигнал может считаться гармоническим сигналом с амплитудной модуляцией. По какому закону, и с какой частотой меняется амплитуда итогового сигнала?

Сохраните проект.

Задание 4.4. Компиляция программы в исполняемый exe-файл

Выберите пункт меню *Tools/Build Application (EXE)* from VI, и подтвердите или введите необходимую информацию в появляющихся диалоговых окнах. – Следует нажать кнопку *Continue*, затем кнопку *Build*. После компиляции *(Build*) программы появляется диалоговое окно, в котором имеется кнопка *Explore* (Исследовать). Нажатием на кнопку *Explore* можно открыть папку с exe-файлом и запустить этот файл.

Задание 4.5. Измерение эффективной величины и амплитуды сигналов разной формы

Создайте новый проект и добавьте в проект виртуальный генератор с необходимыми элементами регулировки, а также добавьте на блок-схему компонент *Express/ Signal Analysis/ Amplitude* and *Level Measurements* (измерение амплитуды и уровня сигнала). Он выдает на выход *RMS (Root Mean Square)* среднеквадратичное значение входного сигнала (то есть эффективное значение по постоянному току), если в свойствах установлен флажок *RMS* - в противном случае выход *RMS* не показывается.

Аналогично, установка флажка *DC* (сокращение от *direct current* – постоянный электрический ток) приводит к появлению выходного контакта *Mean (DC),* на который выводится среднее значение сигнала по постоянному току (после усреднения за относительно большой промежуток времени). Установка флажка *Maximum Peak* – контакта *Positive Peak*, на который выводится амплитудное значение сигнала в положительной области сигнала, и т.д.

Измерьте эффективное, амплитудное и среднее значение сигнала, подаваемого с генератора, для сигналов разной формы.

Сохраните проект.

Задание 4.6. Расчет электрической цепи с потенциометром

Воспользуйтесь сохраненным ранее проектом со схемой электрической цепи из *Лабораторной работы №1*, *Задание 3.8 Расчет электрической цепи с потенциометром.* Создайте копию проекта.

В этой схеме напряжение **E** подается с источника напряжения на потенциометр, имеющий максимальное сопротивление между контактами $\mathbf{R_{max}}$. Измеряется напряжение $\mathbf{U_{вых}}$ между ползунком потенциометра и “землей”.

Напишите формулу зависимости выходного напряжения с потенциометра $\mathbf{U_{вых}}$ от положения ползунка (сопротивления **R** между ползунком и заземленным выводом потенциометра).

Создайте виртуальный прибор для расчета электрической цепи с потенциометром. Поместите на лицевую панель необходимые элементы ввода/вывода, присвойте им имена. Воспользуйтесь узлом Formula или Formula Node для расчета выходного напряжения $\mathbf{U_{вых}}$:

а) без учета сопротивления вольтметра $\mathbf{R_v}$,

б) с учетом сопротивления вольтметра $\mathbf{R_v}$.

Запустите программу на исполнение. Убедитесь в том, что в данном случае использование узла *Formula Node* значительно проще, чем построение

соответствующей программы с использованием элементов *Arithmetic&Comparison*, выполняющих отдельные математические операции.

Сохраните проект.

Задание 4.7. Генератор случайных чисел

4.7.1. Непрерывная генерация случайных чисел

В этом задании изучается, как с помощью средств LabView генерировать и наблюдать на дисплее виртуального осциллографа последовательность случайных чисел.

Создайте новый проект. Установите на лицевой панели прибора графический индикатор (*Controls/Express/Graphic Indicators/Waveform Chart*).

Перейдите в окно *Block Diagram*. Создайте цикл *While Loop (Express/Exec Control/While Loop*).

Разместите внутри рамки цикла генератор случайных чисел, который изображается в виде двух игральных кубиков (рис.2.11), расположенный в разделе *Functions/Programming/Numeric/Random Number (0-1)* (также можно использовать путь *Express/Arithmetic&Comparison/Numeric/Random Num).*

Данный компонент позволяет генерировать псевдослучайные числа в пределах от 0 до 1. Числа называются псевдослучайными потому, что их последовательность хоть и похожа на случайную, но генерируется на основе алгоритма, позволяющего при необходимости воспроизводить одну и ту же последовательность таких чисел.

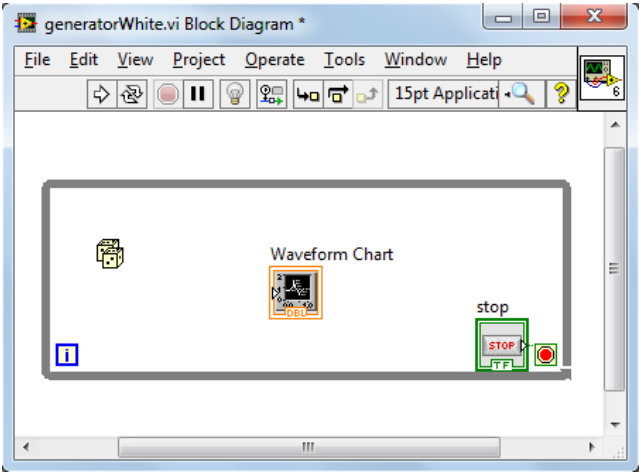

Рис.2.11. Вид *Block Diagram* после добавления компонента *Random Numbers (0-1).*

Для того чтобы случайные числа лежали в диапазоне от **a** до **b**, необходимо умножить их на константу **k = b-a**, и добавить другую константу, равную **a**.

Выберите из раздела *Functions/Express/Arith&Comparison/Numeric* (либо *Express/Arithmetic&Comparison/Numeric)* компоненты *Add u Multiply*, а также две константы *Num Const* из этого же раздела. Первая из них будет использоваться в качестве **k**, вторая – в качестве **a**.

Присвойте константам числовые значения, например 155 и 100.

Соедините все элементы на *Block Diagram* так, как это показано на рис.2.12. Теперь сгенерированные числа будут находиться в диапазоне от 100 до 255, и их значения будут выводиться в компонент построения графика.

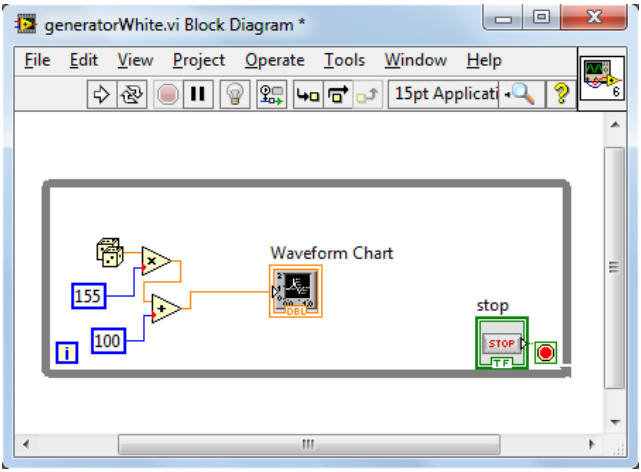

Рис.2.12. Вид блок-схемы программы вывода случайных чисел на график.

После соединения выхода компонента *Add* со входом *Waveform Chart* получаем программу, которая генерирует случайные числа и выводит их на график (рис.2.13). Запустите программу и проверьте ее работу.

Сохраните проект.

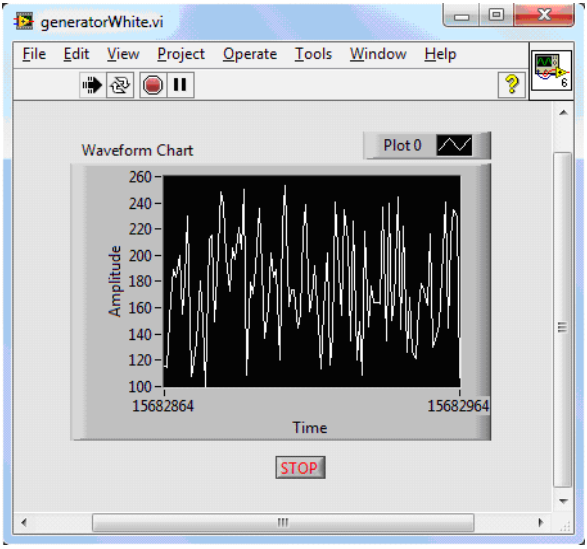

Рис.2.13. Работа программы вывода случайных чисел на график.

4.7.2. Генерация псевдослучайного целого числа по нажатию кнопки. Обработка событий с помощью Event Structure

Сделайте копию предыдущего проекта. Измените программу так, чтобы она выводила в поле индикатора случайное число по нажатию на кнопку. Для этого уберите с лицевой панели предыдущего проекта элемент Waveform Chart. Установите на лицевую панель кнопку *Express/Buttons/Ok Button* и поле текстового вывода чисел *Controls/Express/Num Inds/Num Ind (*рис.2.14).

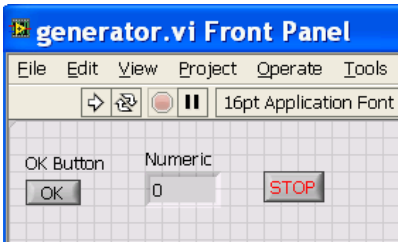

Рис.2.14. Интерфейс программы после расположения числового индикатора и кнопки.

Затем необходимо сменить подписи компонентов (рис.2.15).

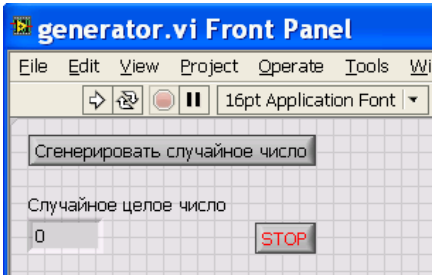

Рис.2.15.Интерфейс программы после смены подписей на компонентах.

Текст на компоненте *OK Button* можно менять прямо на кнопке. Для того чтобы рядом с ним не высвечивался текст “*OK Button*”, как на рис.2.14, необходимо зайти в окно редактирования свойств компонента (рис.2.16) и убрать галочку для свойства *Visible* метки (*Label*) компонента. Убирать текст в самой метке неправильно – в этом случае вы задаете у компонента пустое имя. При этом тексту, который написан на самой кнопке, соответствует свойство *Off text* (“текст при не нажатой кнопке”).

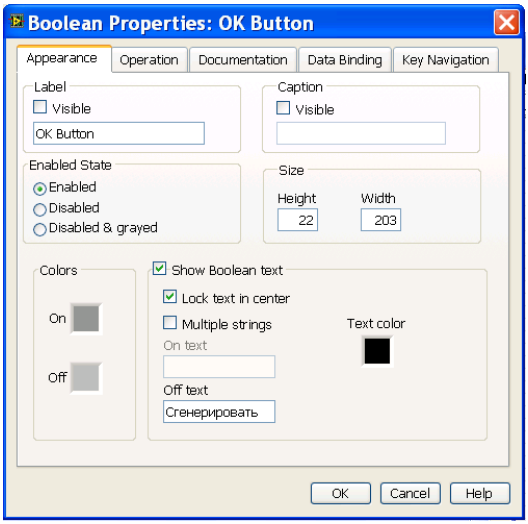

Рис.2.16. Редактирование свойств компонента *OK Button*.

Если попытаться изменить подпись около компонента *Numeric*, будет изменено свойство *Label* (“метка”) – имя компонента (в других системах программирования такое свойство обычно называют *Name*). Оно отображается как на блок-схеме, так и на лицевой панели рядом с компонентом как его подпись, если в поле *Visible* в свойствах компонента стоит галочка. Это имя может быть задано на произвольном национальном языке, в том числе на русском.

Однако использование таких имен не всегда оправданно – в реальных проектах место на блок-схеме приходится экономить, и имя “Случайное целое число” для компонента *Numeric* не очень уместно. Поэтому для подписи около компонента *Numeric* лучше использовать свойство *Caption* (“надпись”). Для этого сначала следует убрать галочку в поле *Visible* в разделе свойства *Label* и поставить галочку в поле *Visible* в разделе свойства *Caption*. После этого можно редактировать текст в разделе свойства *Caption*. Он будет отображаться на лицевой панели в качестве подписи к компоненту.

Получившийся код программы показан на рис.2.17.

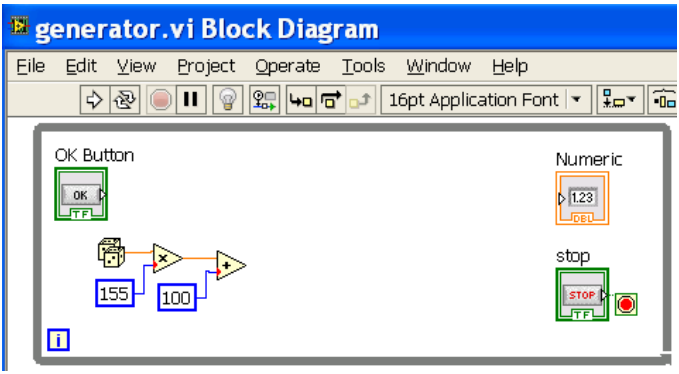

Рис.2.17. Вид программы перед созданием обработчика события.

Реакция на то или иное событие (event) обеспечивается с помощью так называемых обработчиков событий. Например, нажал пользователь на первую кнопку – выполнился алгоритм номер один. Нажал на вторую кнопку – выполнился алгоритм номер два. И так далее. Каждому событию, которое необходимо обработать, сопоставляется свой алгоритм, который запускается при наступлении этого события. В нашем случае это будет генерация очередного числа по нажатию на кнопку *OK Button*. Для создания обработчика события установите на блок-схему компонент *Event Structure* (*Functions/Programming/Structures/Event Structure*) и поместите рамку структуры в пустой области внутри основного цикла программы (рис.2.18).

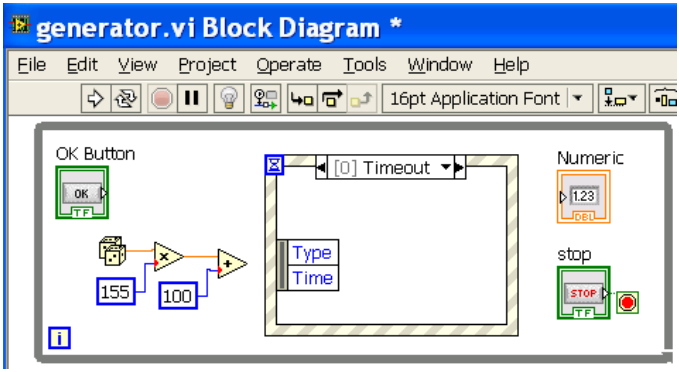

Рис.2.18. Начало создания обработчика события.

По умолчанию создается вкладка *Timeout*, алгоритм внутри которой выполняется во время ожидания события. Необходимо указать время ожидания до следующей итерации цикла в миллисекундах, например 100 мс, подключив целочисленную константу 100 к входу «песочные часы» в левом верхнем углу рамки. В течение этого времени будет происходить ожидание события и выполняться алгоритм из вкладки *Timeout*. Если за время таймаута событие не наступит, структура закончит работу.

Для добавления на рамку новой вкладки, предназначенной для обработки события нажатия на кнопку, необходимо щелкнуть правой кнопкой мыши по рамке Event Structure и выбрать *Add Event Case* (добавить вариант события) – рис.2.19.

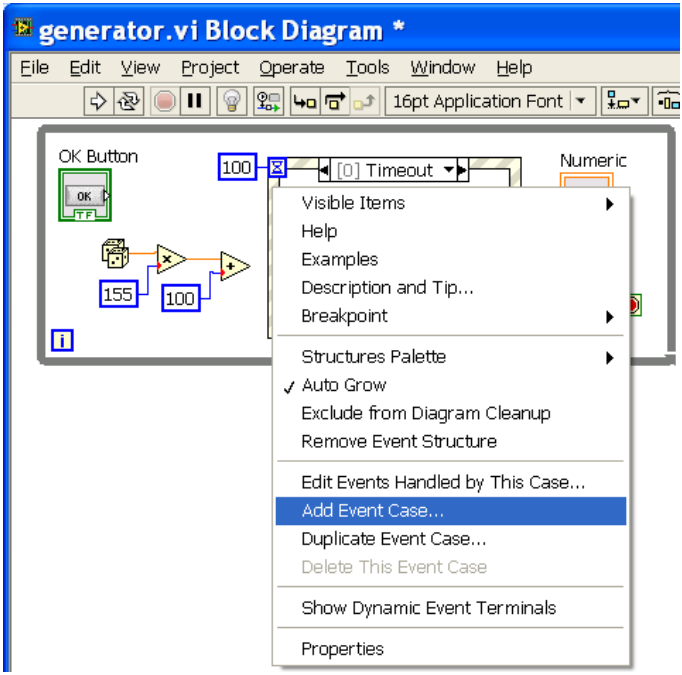

Рис.2.19. Начало добавления обработчика события нажатия на кнопку.

В появившемся окне с деревом элементов (рис.2.20) следует выбрать необходимую кнопку (*OK Button*) и ее событие *Value Change*, и нажать кнопку *OK*.

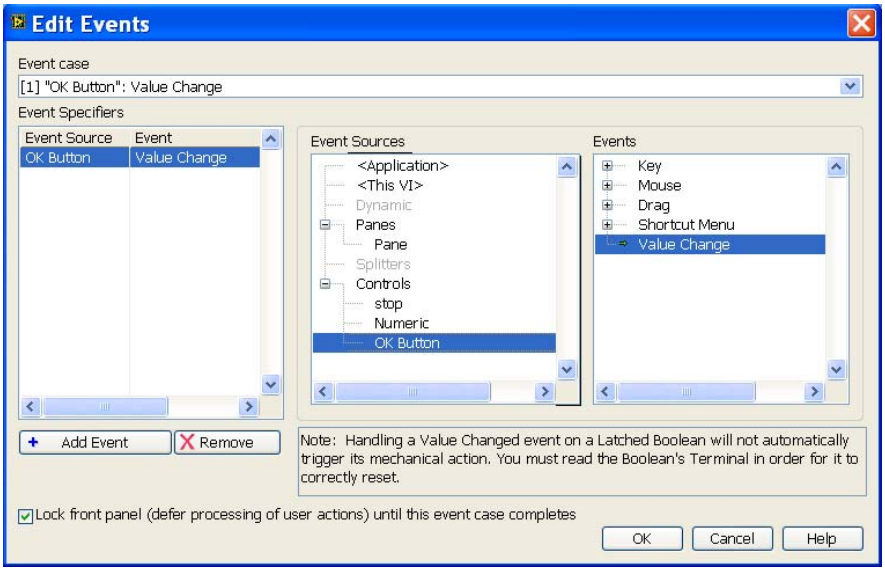

Рис.2.20. Добавление обработчика события нажатия на кнопку.

После чего необходимо выделить мышкой прямоугольник, в котором находятся элементы генерации случайного числа, и перетащить их внутрь появившейся в структуре вкладки "*OK Button": Value Change* (рис.2.21).

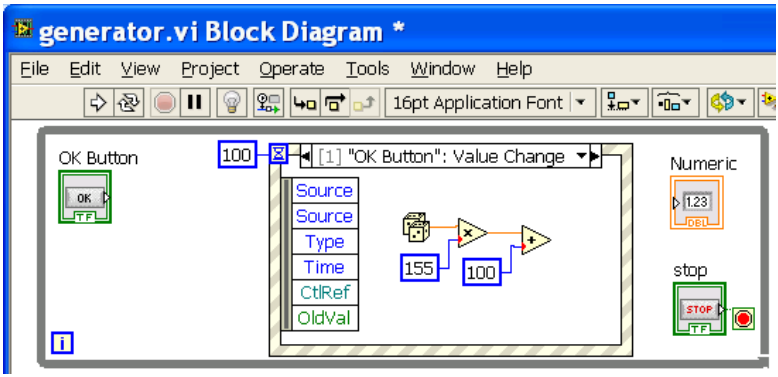

Рис.2.21. Добавление в *Event Structure* обработчика события.

Для сохранения полученного случайного числа между итерациями цикла *While* (на рис.2.21 блок цикла обозначен внешней сплошной серой рамкой) необходимо переключиться во вложенной в нее *Event Structure* во вкладку

Timeout нажатием стрелки влево или стрелки вправо в заголовке рамки. Содержание этой вкладки показано на рис. 2.18.

Для правильной работы вход и выход цикла *While* должны быть сконфигурированы как сдвиговый регистр (*Shift Register*) – то, что поступает на вход, должно приходить на выход. Щелкните правой кнопки мыши на *левой* границе рамки цикла *While Loop* и выберите в появившемся меню пункт *Add Shift Register* (рис.2.22*)*.

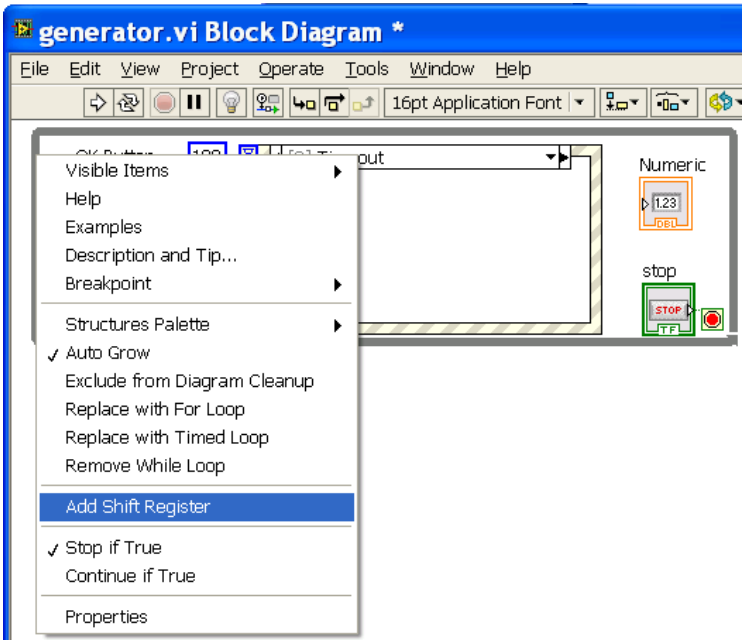

Рис.2.22. Начало настройки передачи данных в следующую итерацию цикла.

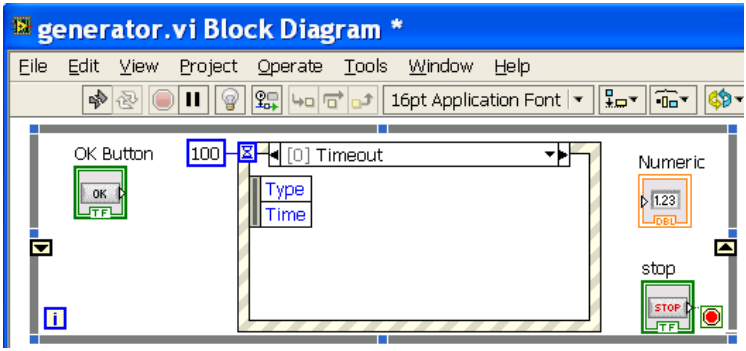

Рис.2.23.Настройка передачи данных в следующую итерацию цикла.

На левой части рамки возникнет входной терминал для данных цикла *While* в виде стрелки вниз (“войти внутрь”), а на правой – выходной терминал для данных в виде стрелки вверх (“выйти наружу”) (рис.2.23). Протяните провод от входного терминала цикла *While* до рамки левой границы *Event Structure* и щелкните по ней – и на этой границе возникнет терминал в виде черного квадратика. На него будет приходить входные данные *Event Structure*, поступающие на обработку внутри этого элемента. Протяните от этого квадратика провод до стрелки на правой границе цикла *While* (выходные данные цикла). При этом на правой границе рамки *Event Structure* терминал возникнет автоматически (рис.2.24).

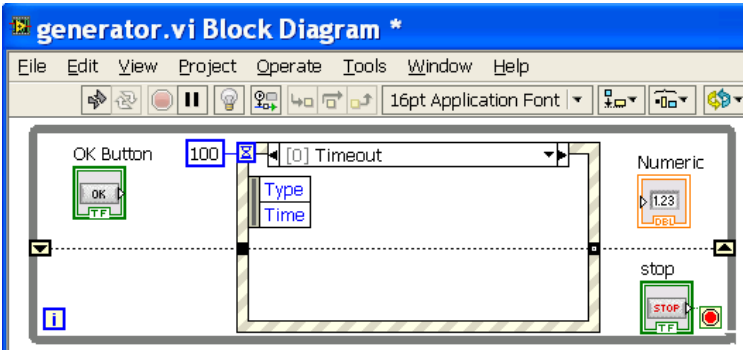

Рис.2.24.Настройка передачи данных в следующую итерацию цикла.

Получившиеся линии служат для передачи данных при работе основного цикла работы программы в том случае, когда нет события. Через каждые 100

мс проходит проверка на наличие события нажатия на кнопку, и при отсутствии события данные со входа основного цикла программы без изменения внутри рамки *Event Structure* передаются на выход.

Для того чтобы подключить код генерации чисел, срабатывающий при нажатии на кнопку, переключитесь во вкладку "*OK Button": Value Change*. Соедините выход генератора случайных чисел к точке выхода *Event* Structure, а далее – к индикатору, показывающему значение числа (рис.2.25). В результате подсоединения генератора, на выходе которого числа *Double*, терминалы и провода приобрели оранжевый цвет.

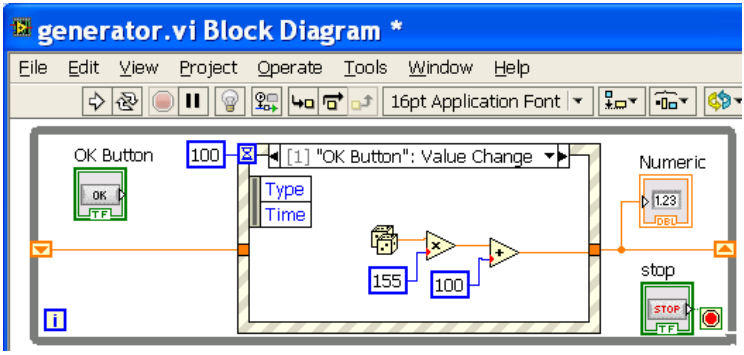

Рис.2.25.Завершение настройки передачи данных в следующую итерацию цикла.

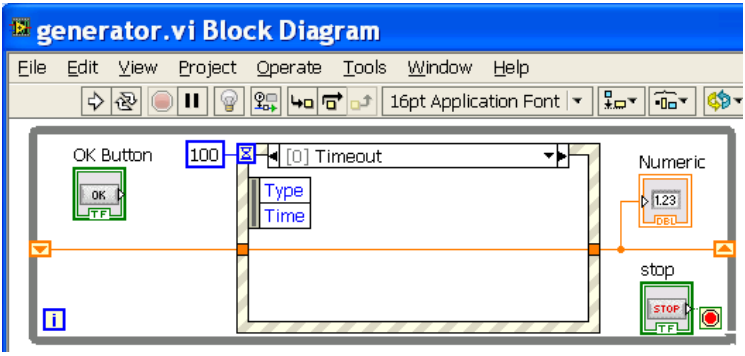

Рис.2.26. Передача данных от предыдущей итерации цикла к следующей.

При нажатии на кнопку сгенерированное значение будет подано на выход цикла *While* и индикатор (рис.2.25). В следующей итерации цикла *While*

нажатия на кнопку уже нет, и реализуется ситуация, которую можно увидеть, переключившись на вкладку *Timeout* (рис.2.26).

Данные проходят от входа цикла *While* к его выходу, проходя через блок *Event Case* на вкладке *TimeOut* без изменений. Тем самым в отсутствие действий пользователя значение, подаваемое на индикатор, будет сохраняться – то есть индикатор будет отображать те значения, которые были получены при нажатии кнопки.

Компонентом *Random Number (0-1)* генерируются вещественные числа, содержащие дробную часть. Но если сигнал с выхода генератора подать на компонент, показывающий целочисленные значения, поступающие на него числа будут сначала округляться до целого и только потом показываться. Поэтому для показа округленных значений достаточно установить целочисленный тип индикатора. Но если числа с генератора должны поступать куда-то еще, и они должны быть целыми, их требуется округлить. Сделаем это с помощью компонента *Functions/ Programming/* Numeric/ *Round To Nearest.* Переключитесь во вкладку *"OK Button": Value Change,* перетащите его на схему поверх выходного провода с генератора – и он автоматически встроится в схему (рис.2.27).

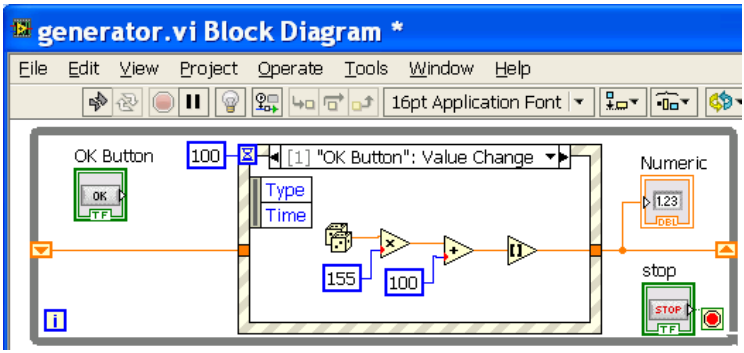

Рис.2.27. Вывод округленных значений.

На первый взгляд программа написана и работает правильно, но это не совсем так. В проведенном выше анализе работы программы был упущен

момент начального состояния программы до того, как в первый раз будет нажата кнопка. Какое значение будет передаваться с входа на выход цикла?

В LabView первоначально будет 0, а при последующих запусках будет показываться последнее значение из предыдущего запуска. Для того чтобы при запуске программы на вход цикла ожидания всегда поступало одно и то же значение, которое и будет показываться на индикаторе до первого нажатия кнопки генерации случайного числа, следует создать числовую константу и присоединить ее снаружи к входному терминалу цикла (рис.2.28). В нашем случае естественно использовать значение 0.

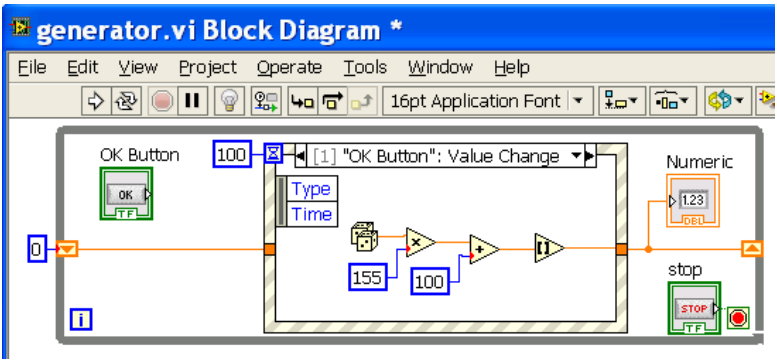

Рис.2.28.Окончательный вариант программы.

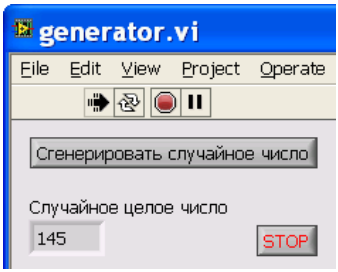

Рис.2.29.Пример работы программы.

На рис.2.29 показана работа получившейся программы. Сохраните проект. Запустите программу и проверьте ее работу, в том числе при повторных запусках как при отсутствии входной константы, так и при ее наличии.

4.7.3. Создание одномерного массива псевдослучайных чисел

Обрабатывать нажатие на кнопку можно и по изменению булевского значения на выходе кнопки: когда кнопка не нажата, на ее выходе значение *False*, а при нажатой кнопке на ее выходе *True*.

Используйте копию полученной в *Задании 4.7.2* программы для создания по нажатию кнопки одномерного массива случайных чисел и сохранения его в таблице (Table). Для этого переключитесь во вкладку "*OK Button": Value Change* (рис.2.28). Выделите мышью прямоугольник с элементами генератора и перетащите их вне основного цикла программы. После чего удалите рамку *Event Structure*, перетащите элементы генератора обратно в основной цикл программы, и удалите все лишние элементы так, чтобы получилась блок-схема, показанная на рис.2.30.

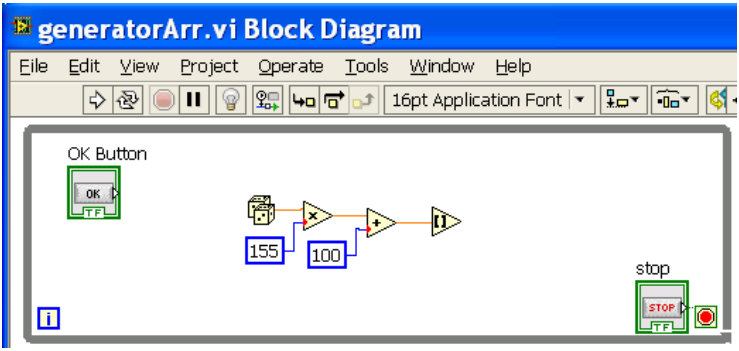

Рис.2.30. Вид блок-схемы после удаления лишних компонентов.

Измените надпись на кнопке на "Сгенерировать массив случайных чисел". Затем добавьте в предыдущую программу на лицевую панель таблицу *Express/Text Indicators/Table*. Ограничимся одномерным массивом из десяти чисел – правой кнопкой мыши вызовите установку свойств таблицы и задайте число строк (*Rows)* равным 10, а число столбцов равным 1. При этом лицевая панель приобретет вид, схожий с показанным на рис.2.31.

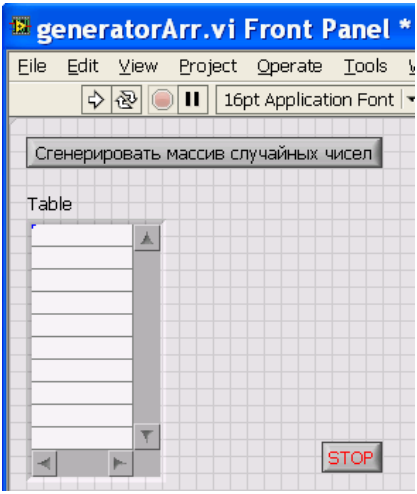

Рис.2.31. Добавление таблицы на лицевую панель.

На блок-схеме появятся два новых компонента, соединенные друг с другом – *Build Table* (“построить таблицу”) и Table (рис.2.32.)

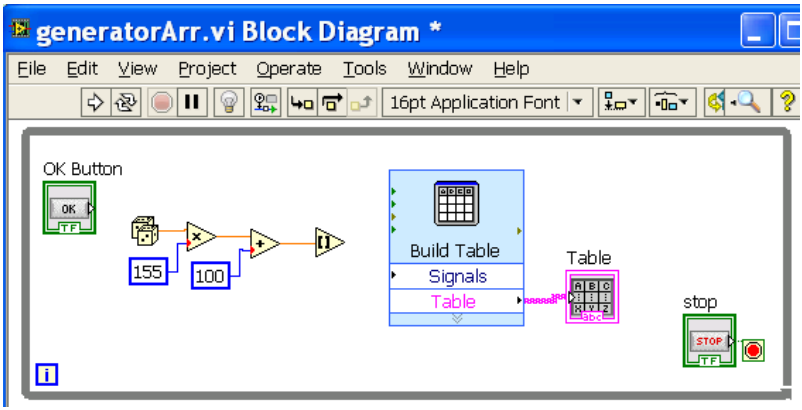

Рис.2.32. Блок-схема программы после добавления на лицевую панель таблицы.

Сигнал с выхода построенного ранее генератора псевдослучайных чисел необходимо подать на вход *Signals* компонента *Build Table* (рис.2.33).

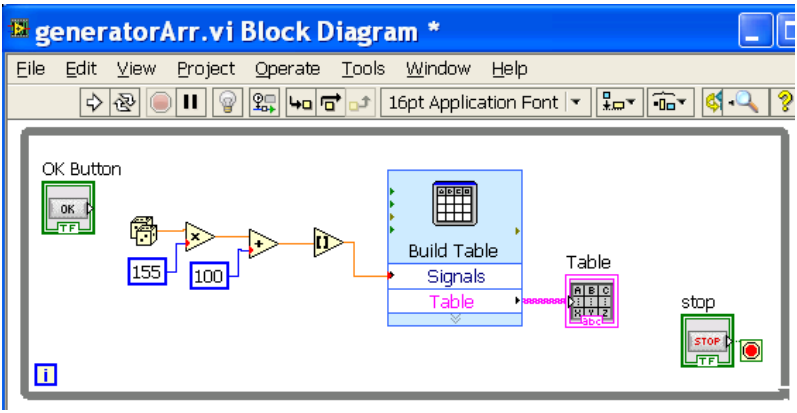

Рис.2.33. Вид блок-схемы после добавления на лицевую панель таблицы.

Для получения массива надо десять раз выполнить генерацию чисел – воспользоваться циклом *For Loop (Functions/ Programming/ Structures/ For Loop*). Обведите рамкой структуры ту часть схемы, которая должна работать в цикле – получится схема, аналогичная показанной на рис.2.34. Можно обводить не целый компонент, а только его часть, если рядом находится элемент, который не должен попасть в рамку – все равно компонент окажется выделенным и попадет внутрь рамки.

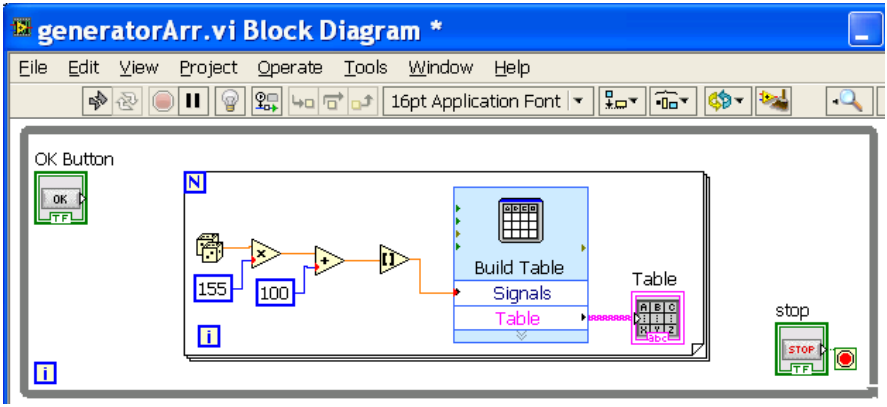

Рис.2.34. Добавление цикла *For Loop* в программу.

В отличие от использовавшегося ранее цикла *While Loop* цикл *For Loop* выполняется заданное число раз, устанавливаемое через вход числа операций

N (показывается в левом верхнем углу рамки). Как и для цикла *While Loop* блок **i** в левом нижнем углу – это счетчик числа операций (счетчик цикла).

В нашем примере число операций задается как число элементов массива на выходе цикла. Для управления количеством повторений подключите числовую константу 10 к блоку числа операций N в левом верхнем углу компонента цикла *For Loop.*

Далее для проверки того, нажата ли кнопка, следует добавить компонент *Case Structure (Functions/Programming/Structures/Case Structure)* – поместить рамку цикла *For Loop* внутрь вкладки *True* компонента *Case Structure*. После чего выход кнопки необходимо подключить к входу *Case Structure* (рис.2.35).Сохраните проект.

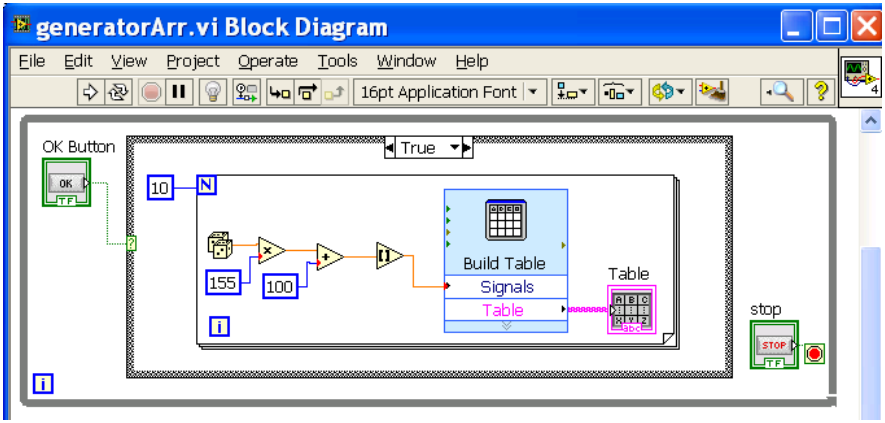

Рис.2.35. Схема запуска цикла нажатием на кнопку.

Работа программы представлена на рис.2.36.

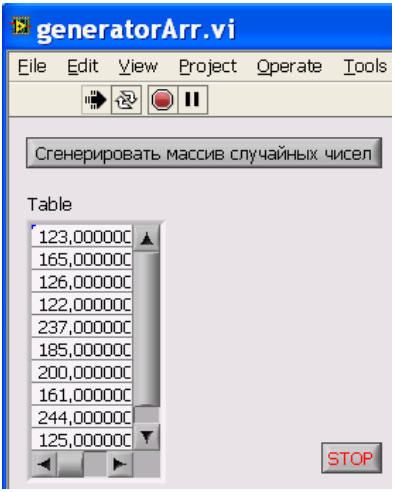

Рис.2.36.Пример работы программы.

Для того чтобы в таблице показывались целые числа, без нулей после десятичного разделителя, необходимо зайти на блок-схеме в параметры компонента *Build Table* и установить в качестве типа показываемой в таблице величины десятичное целое с 12 десятичными разрядами *Decimal Integer* (12) – рис.2.37.

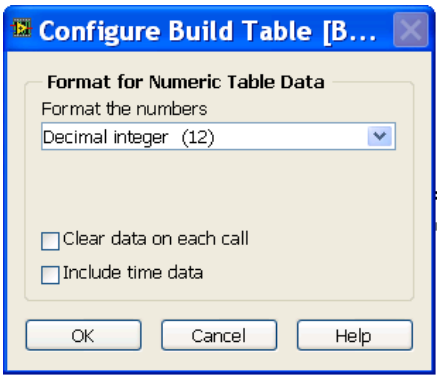

Рис.2.37.Установка типа показываемой в таблице величины.

Полученная программа имеет особенность – при повторных нажатиях кнопки в таблице добавляются новые строки после старых. Но иногда требуется заменить выведенные ранее данные. У компонента *Build Table*

имеется вход *Reset* (“переустановить”) для сброса таблицы в начальное состояние – как это сделать показано на рис. 2.38.

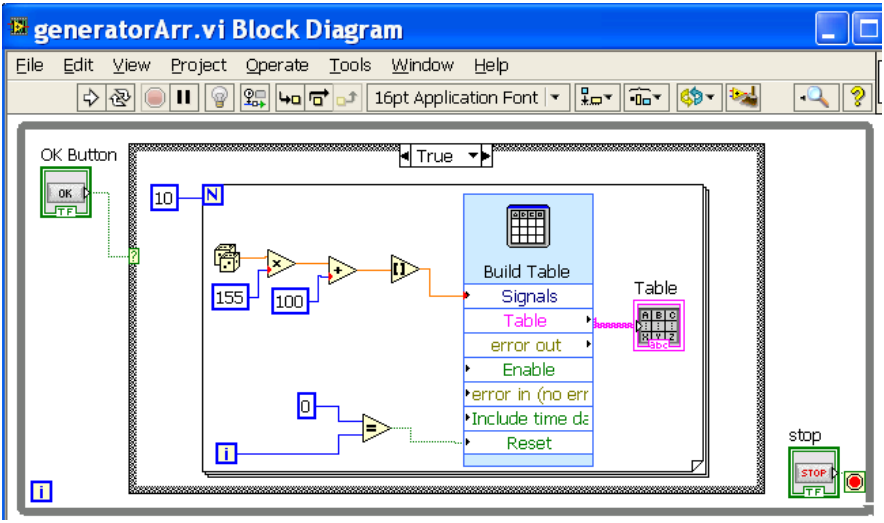

Рис.2.38. Схема запуска цикла и сброса таблицы нажатием на кнопку.

III. Лабораторная работа № 3-NI. Программирование в среде LabView цифровых портов ввода-вывода устройства сбора данных NI myDAQ

1. Теоретическая часть

1.1. Позиционные системы счисления

Позиционная система счисления - это такой способ записи числа, при котором вес цифры зависит от занимаемой позиции и пропорционален степени некоторого числа. Основание степени называется основанием системы счисления.

Наиболее распространены десятичная (основание десять), шестнадцатеричная (основание шестнадцать), восьмеричная (основание восемь) и двоичная (основание два) системы. Число различных знаков - цифр, используемых для записи чисел – в каждой системе равно основанию данной системы счисления.

0,1	- цифры двоичной системы
0,1,2,3,4,5,6,7	- цифры восьмеричной системы

0,1,2,3,4,5,6,7,8,9 - цифры десятичной системы
0,1,2,3,4,5,6,7,8,9,A,B,B,C,D,E,F - цифры шестнадцатеричной системы

В шестнадцатеричной системе арабских десятичных цифр недостаточно, и для обозначения цифр больших девяти, используются заглавные латинские буквы A,B,C,D,E,F.

В дальнейшем везде, где это необходимо, мы будем указывать основание системы счисления индексом рядом с числом: 126_{10} - в десятичной системе, $7E_{16}$ – в шестнадцатеричной системе, 176_8 - в восьмеричной системе, 1111110_2 – в двоичной системе.

Применительно к компьютерной обработке данных под информацией понимают некоторую последовательность символических обозначений (букв, цифр, закодированных графических образов и т.д.), несущую смысловую нагрузку и представленную в виде, который может быть обработан какой-либо компьютерной программой.

Двоичная система является естественным способом кодирования информации в компьютере, когда сообщение представляется набором нулей (0 - нет сигнала на линии) и единиц (1 - есть сигнал на линии). Процессор компьютера обрабатывает информацию, представленную именно в таком виде. Для обозначения двоичных цифр применяется термин “бит”, являющийся сокращением английского словосочетания “двоичная цифра” (BInary digiT). Архитектура компьютера накладывает существенное ограничение на количество информации, обрабатываемой устройством за одну операцию. Оно измеряется числом двоичных разрядов и называется *разрядностью* этого устройства.

С помощью восьми двоичных разрядов можно представить 2^8= 256 целых чисел. Порция информации размером 8 бит (восьмибитовое число) служит основной единицей измерения компьютерной информации и называется байтом (byte). Как правило, передача информации внутри компьютера и между компьютерами идет порциями, кратными целому числу байт. *Машинным словом* называют порцию данных, которую процессор компьютера может обработать за одну операцию (микрокоманду). Первые

персональные компьютеры (PC - personal computer) были 16-разрядными, т.е. работали с 16-битными (двухбайтными) словами. Поэтому операционные системы первых поколений для этих компьютеров также были 16-разрядными. Например, MS DOS. В дальнейшем операционные системы стали 32-разрядными, так как были ориентированы на использование 32-разрядных процессоров. В настоящее время чаще всего используются 64-разрядные версии Windows, Mac OS и Linux для 64-разрядных процессоров.

Шестнадцатью битами можно представить числа от 0000000000000000_2 до 1111111111111111_2 (от 0 до $2^{16}-1 = 65535$). Аналогично, 32-битное слово может быть использовано для представления чисел в диапазоне от 0 до $2^{32}-1$, а 64-битное – для представления чисел от 0 до $2^{64}-1$.

Представление чисел в двоичной и шестнадцатеричной системах счисления, а также преобразование из одной системы в другую бывает необходимо при программировании аппаратуры, предназначенной для измерений и управления различными устройствами. Эти действия осуществляются с помощью портов ввода-вывода, цифро-аналоговых и аналого-цифровых преобразователей. Компьютеры на основе процессоров x86 (от Intel i-86 до Pentium Duo, Xeon и т.д.; AMD Athlon, Opteron и т.д.) имеют специальные адреса для управления аппаратурой - адреса портов. Обращение к портам независимо от разрядности процессора и типа операционной системы может производиться либо побайтно (как по четным, так и по нечетным адресам портов), либо 16-разрядными словами (только по четным адресам портов).

1.2. Преобразование чисел из одной позиционной системы счисления в другую

Число N может быть записано с помощью разных систем счисления. Например, в десятичной:

$$N = A_n \cdot 10^n + \ldots + A_2 \cdot 10^2 + A_1 \cdot 10^1 + A_0 \cdot 10^0, \ (A_k = 0 \ldots 9)$$

или в двоичной:

$$N = B_p \cdot 2^p + \ldots + B_2 \cdot 2^2 + B_1 \cdot 2^1 + B_0 \cdot 2^0, \ (B_k = 0 \text{ или } 1)$$

Преобразование в другую систему счисления сводится к нахождению коэффициентов B_k по известным коэффициентам A_m при переводе из десятичной системы в двоичную, или коэффициентов A_m по коэффициентам B_k при переводе из двоичной системы в десятичную.

1.2.1. Преобразование чисел из системы с большим основанием в систему с меньшим основанием

Рассмотрим преобразование из десятичной системы в двоичную. Для известного числа N необходимо найти коэффициенты в выражении

$$N = B_n \cdot 2^n + \ldots + B_2 \cdot 2^2 + B_1 \cdot 2^1 + B_0 \cdot 2^0, \ (B_n = 0 \text{ или } 1)$$

Воспользуемся следующим алгоритмом: в десятичной системе разделим число N на 2 с остатком. Остаток деления (он не превосходит делителя) даст коэффициент B_0 при младшей степени, то есть при 2^0. Далее делим на 2 частное, полученное от предыдущего деления. Остаток деления будет следующим коэффициентом B_1 двоичной записи N. Повторяя эту процедуру до тех пор, пока частное не станет равным нулю, получим последовательность коэффициентов B_n.

Если преобразуемое число невелико (например, $N \leq 1024=2^{10}$) можно легко найти коэффициенты $B_{n,}$, начиная с коэффициента при максимальной степени двойки, содержащейся в данном числе.

Пример: Преобразуем 345_{10} к двоичному виду. Имеем

$$345_{10} = 1 \cdot 2^8 + 0 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 101011001_2$$

1.2.2. Преобразование чисел из системы с меньшим основанием в систему с большим основанием

1.Рассмотрим преобразование из двоичной системы в десятичную. Запишем число N в виде

$$N = B_p \cdot 2^p + \ldots + B_2 \cdot 2^2 + B_1 \cdot 2^1 + B_0 \cdot 2^0 \ (B_k = 0 \text{ или } 1)$$

и будем рассматривать эту формулу как алгебраическое выражение в десятичной системе. Выполним арифметические действия по правилам десятичной системы. Полученный результат даст десятичное представление числа N.

Пример: Преобразуем 01011110_2 к десятичному виду. Имеем:

$$01011110_2 = 0 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 =$$

$$0 + 64 + 0 + 16 + 8 + 4 + 2 + 0 = 94_{10}$$

2. Рассмотрим преобразование из десятичной системы в шестнадцатеричную. Заметим, что $16=2^4$. Преобразуем десятичное число в двоичное: объединим цифры в двоичной записи числа группами по четыре (справа налево), и каждую такую группу (тетраду) заменим соответствующей шестнадцатеричной цифрой. Таким образом, перевод чисел из двоичного представления в шестнадцатеричное и обратно осуществляется простой заменой всех групп из четырех двоичных цифр на шестнадцатеричные цифры:

$0000_2 = 0_{16}$	$0100_2 = 4_{16}$	$1000_2 = 8_{16}$	$1100_2 = C_{16}$
$0001_2 = 1_{16}$	$0101_2 = 5_{16}$	$1001_2 = 9_{16}$	$1101_2 = D_{16}$
$0010_2 = 2_{16}$	$0110_2 = 6_{16}$	$1010_2 = A_{16}$	$1110_2 = E_{16}$
$0011_2 = 3_{16}$	$0111_2 = 7_{16}$	$1011_2 = B_{16}$	$1111_2 = F_{16}$

Пример: Преобразуем 1011010111_2 к шестнадцатеричному виду:

$$10\ 1101\ 0111_2 = 0010\ 1101\ 0111_2 = 2D7_{16}$$

1.3. Порты цифрового ввода - вывода

Общение компьютера с различными подключаемыми к нему устройствами, как уже говорилось, обычно происходит через так называемые порты (“порт” в переводе с греческого – “ворота”). Для идентификации устройств ввода-вывода в персональном компьютере каждому из них присвоен индивидуальный номер – “адрес порта”. При каждой операции записи/чтения происходит обращение только к одному устройству - тому, адрес которого совпадает с заданным.

Данные, записанные в порт, сохраняются в специальном буфере (регистре порта) до следующей записи данных в этот порт. При операции “чтение из порта” происходит вывод цифровой информации о состоянии порта в момент операции чтения. Порты чтения и записи с одинаковыми адресами могут быть физически разделены, при этом часто данные, записанные в порт, не

могут быть прочитаны при обращении к этому порту. В PC старых образцов существовали два специальных типа устройств ввода-вывода, также называемых портами - 1) последовательные 2) параллельные.

Последовательные порты (обычно их было два - COM1 и COM2) раньше служили для подключения “мыши” и других относительно медленных устройств. В настоящее время они заменены USB-портами (сокращение от Universal Serial Bus – “универсальная последовательная шина”). В последовательных портах передача данных происходит по одному проводу бит за битом (как правило, порциями по 1 байту, с добавлением 1-2 служебных битов).

В параллельных портах передача данных происходит одновременно по 8 проводам байт за байтом. Параллельные порты PC в настоящее время также практически вытеснены USB-портами. Тем не менее, при управлении устройствами автоматизации часто приходится использовать параллельные порты: для этих целей выпускаются специальные электронные модули, подключаемые к компьютеру через USB или другим способом, и имеющие внешние разъемы параллельных портов ввода-вывода.

2. Компьютерный тест №3

В тесте проверяется усвоение учащимся теоретической части описания лабораторной работы. Для выполнения теста следует зайти на в систему контроля качества обучения и перейти по ссылке “Компьютерный тест “№3”.

После прохождения теста в разделе «Результаты» правильные ответы помечаются зеленым цветом, неправильные – красным. Результаты прохождения теста служат допуском к выполнению лабораторной работы.

3. Преобразование чисел из одной позиционной системы счисления в другую с помощью виртуального устройства Tester

Задание 3.1. Преобразование чисел из десятичной системы счисления в двоичную и обратно

Воспользовавшись приложением *Tester*, созданным при помощи LabView, переведите произвольные числа из десятичной системы счисления в двоичную, затем из двоичной в десятичную. Убедитесь в правильности своих вычислений.

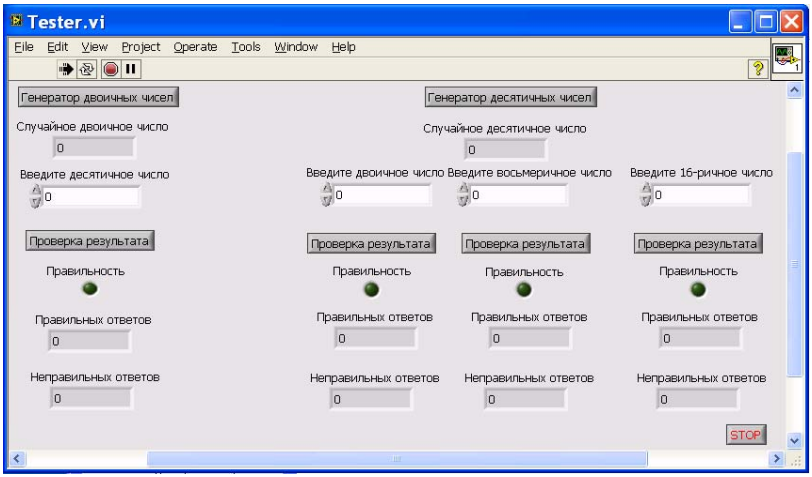

Рис.3.1. Лицевая панель приложения *Tester*.

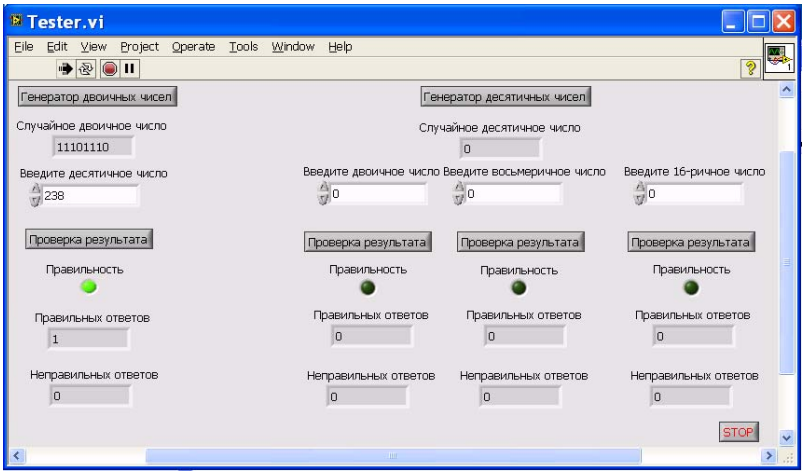

Рис.3.2 . Пример преобразования случайного двоичного числа в десятичное.

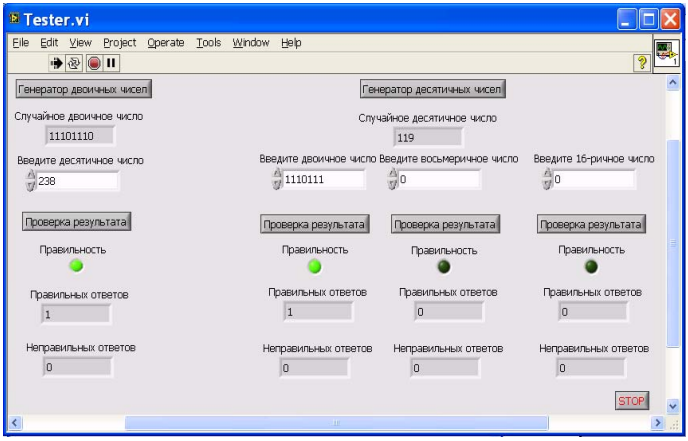

Рис.3.3. Пример преобразования случайного десятичного числа в двоичное.

Задание 3.2. Перевод чисел из десятичной системы счисления в двоичную, восьмеричную и шестнадцатеричную

Переведите произвольно выбранное число из десятичной в двоичную систему, а затем то же число запишите в восьмеричной и шестнадцатеричной системах счисления. Проверьте свой результат.

4. Экспериментальная установка

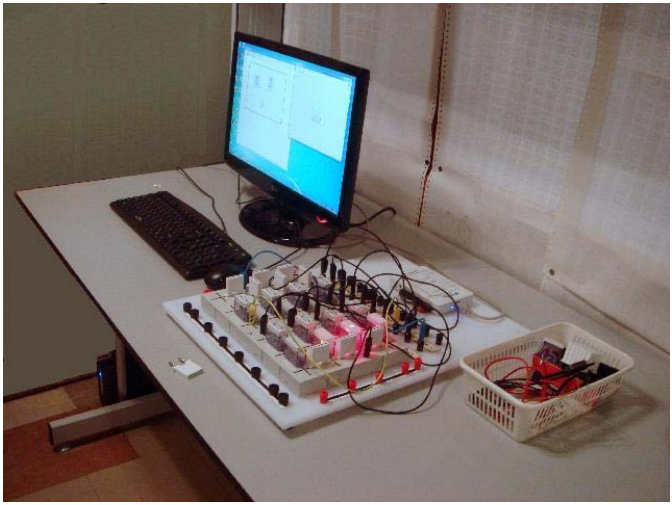

Рис.3.4. Общий вид экспериментальной установки.

Экспериментальная установка для данной лабораторной работы, общий вид которой показан на рис.3.4, состоит из следующих частей:

1) Компьютер.

2) Модуль NI myDAQ (DAQ – Data Acquisition – сбор данных), подключаемый к USB-порту компьютера (рис.3.5).

Рис.3.5. Устройство сбора данных NI myDAQ.

3) Цифровой мультиметр на основе NI myDAC (DMM – Digital MultiMeter). Входы цифрового мультиметра показаны на рис.3.6.

Рис.3.6. Входы цифрового мультиметра на лицевой стороне NI myDAQ.

4) Наборная панель (рис.3.7.) и устанавливаемые на неё элементы (резисторы, перемычки, светодиоды – рис.3.8), провода.

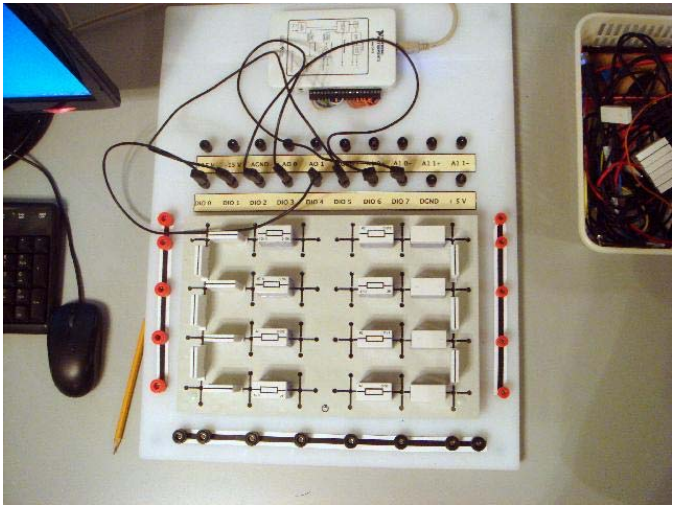

Рис.3.7. Наборная панель с установленными на нее элементами, подсоединенная к NI myDAQ.

Рис.3.8. Элементы наборной панели.

NI myDAQ использует технологию виртуальных приборов NI LabView и позволяет осуществлять сбор и анализ сигналов и управлять простыми процессами в электрических схемах. В устройство NI myDAQ входят два аналого-цифровых преобразователя (АЦП), два цифро-аналоговых преобразователя (ЦАП) и 8 цифровых линий ввода-вывода с возможностью объединения в 8-разрядный параллельный порт, подключаемый через 20-контактный разъем (рис.3.9).

Рис.3.9. Разъемы на боковой стороне NI myDAQ.

В NI myDAQ имеются аналоговые входы (AI – Analog Input) для регистрации аналоговых сигналов с помощью АЦП, аналоговые выходы (A0 – Analog Output) для получения аналоговых сигналов с помощью ЦАП, цифровые входы и выходы (DIO– Digital Input/Output), источники питания, а также вход и выход аудиосигналов.

В NI myDAQ имеется три источника питания: +15 В, -15 В и +5 В. Источники +15 В и -15 В могут быть использованы для питания аналоговых компонентов (например, операционных усилителей, и линейных регуляторов напряжения). Источник +5 В может быть использован для питания цифровых компонентов, таких как логические устройства.

Разъемы AGND и DGND – аналоговая и цифровая земля, соответственно, служат в качестве опорных уровней напряжения для AI, AO ±15 В и DIO и источника +5 В.

Порт NI myDAQ имеет восемь программно тактируемых линий цифрового ввода/ вывода, обозначаемых как *DIO 0 – DIO 7*. Цифровая линия – это путь, по которому передается цифровой сигнал. Линии должны быть индивидуально настроены либо для входа, либо для выхода, они не могут функционировать одновременно в обоих направлениях. Разрядность порта

определяется числом его линий. Цифровые линии NI myDAQ допускают пропускание сигналов логического уровня: напряжение логического нуля от 0 до +0.8 В, напряжение логической единицы от +2.0 В до +5 В.

Для удобства в экспериментальной установке, предлагаемой для выполнения лабораторной работы, винтовые клеммы 20-контактного разъема NI myDAQ соединены проводами с соответствующими клеммами наборной панели, имеющими те же обозначения. С клемм наборной панели сигналы можно подавать в нужные точки собираемой электрической схемы.

После того как NI myDAQ подключен к системе, автоматически запускается виртуальная панель приборов NI ELVISmx (рис.3.10) – оболочка над пакетом созданных в LabView виртуальных приборов, предназначенных для управления реальными приборами.

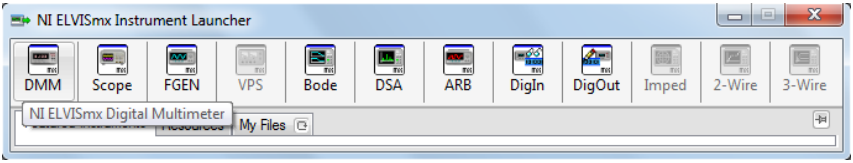

Рис.3.10. Виртуальное устройство NI ELVISmx.

NI ELVISmx предоставляет программные лицевые панели и исходный код для этих виртуальных приборов.

Виртуальный цифровой мультиметр - это программа, которая использует устройство NI myDAQ для измерения напряжений, токов и сопротивлений. Измерения проводятся с помощью имеющихся внутри NI myDAQ аналого-цифровых (АЦП) и цифро-аналоговых (ЦАП) преобразователей. АЦП совершает преобразование имеющегося на его входе аналогового напряжение в пропорциональный этому напряжению двоичный цифровой код, получаемый на выходе и передаваемый в компьютер. ЦАП осуществляет обратное действие – получает из компьютера цифровой код и преобразует его в напряжение на выходе, пропорциональное полученному цифровому коду. Таким образом, при изменении входного кода на выходе ЦАП изменяется аналоговое напряжение. Использовать цифровой

мультиметр можно без знания принципов работы и программирования АЦП и ЦАП, эти устройства будут подробно изучаться в отдельной лабораторной работе.

Цифровой мультиметр DMM (рис.3.11) может выполнять следующие виды измерений: постоянное напряжение (60 В, 20 В, 10 В и 200 мВ); переменное напряжение (20 В, 2 В и 200 мВ); постоянный ток (1 А, 200 мА, 20 мА); переменный ток (1 А, 200 мА, 20 мА), сопротивление (200 МОм, 2 МОм, 200 кОм, 20 кОм, 2 кОм и 200 Ом).

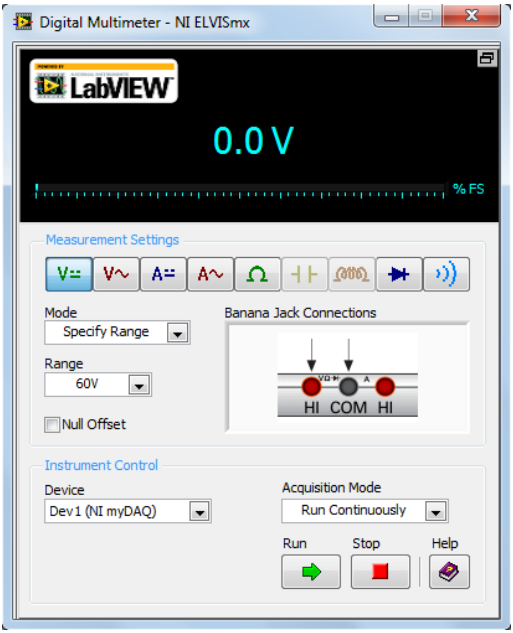

Рис.3.11. Виртуальная панель цифрового мультиметра (DMM).

В данной работе цифровой мультиметр используется для измерения сопротивлений (**Задание 5.1**) и напряжения на выходных клеммах порта перед тем, как подать напряжение на светодиоды (**Задание 5.4***)*.

5. Порядок выполнения работы

При выполнении работы заносите в рабочий журнал и в файл, который прикладывается к отчету, ответы на вопросы во всех заданиях, объяснение результатов, необходимые вычисления и комментарии.

Задание 5.1. Изучение цифрового мультиметра на основе NI myDAQ. Измерение сопротивления

1) Запустите виртуальный мультиметр. Для этого перейдите в меню *Пуск/ Все программы/ National Instruments/ NI ELVISmx for NI ELVIS& NI myDAQ/ NI ELVISmx Instrument Launcher*. Для запуска прибора выберите иконку DMM на лицевой панели. Откроется виртуальная панель цифрового мультиметра (рис.3.11). Для измерения сопротивления необходимо выбрать соответствующий режим работы мультиметра, нажав на кнопку  .

В поле *Range* выберите диапазон сопротивлений в зависимости от номинала резистора, сопротивление которого вам предложено измерить (например, 20Ком,2Ком, 200Ом).

В области "*Banana Jack Connection*" (штырьковые разъёмы) (рис.3.11) отображается схема подключения разъемов. Согласно схеме необходимо подключить черный и красный щупы в соответствующие по цвету гнезда разъёмов на лицевой стороне NI myDAQ (рис.3.6).

Запустите мультиметр, нажав на кнопку *Run*. Какая информация появится на дисплее?

Соедините концы щупов с концами узла, сопротивление которого должно быть измерено. На дисплей выведется значение сопротивления данного резистора.

После измерений прибор необходимо остановить, нажав кнопку *Stop* на лицевой панели

2) Измерьте с помощью цифрового мультиметра сопротивление трех резисторов одинакового номинала. Перейдите на другой диапазон, повторите измерения.

3) Оцените погрешность измерений цифрового мультиметра на разных диапазонах.

4) Сопротивления резисторов обозначаются маркировкой с помощью цветных полосок. Резисторы с одинаковым набором цветных полосок должны обладать одинаковым сопротивлением. Измерьте сопротивления

нескольких резисторов с одинаковыми номинальными сопротивлениями. Оцените отличие измеренных значений сопротивлений от номинальных. Связано ли это отличие с погрешностью измерений цифрового мультиметра?

Результаты всех измерений и вычислений занесите в отчет, сохраните файл.

Задание 5.2. Вывод цифровых сигналов в порт в виде двоичных чисел

В этом задании нужно создать программу, которая отправляет цифровой сигнал на линии порта *DIO 4, DIO 5, DIO 6, DIO 7*. В данном задании эти четыре линии используются в качестве **выходов**, а остальные четыре линии не используются.

***5.2.1**. Написание программы*

Для решения этой задачи в окне *Front Panel* нужно построить лицевую панель виртуального прибора. В качестве элементов управления выходным сигналом используйте 4 тумблера (в окне *Front panel: Controls/Express/ Buttons&Switches/ Toggle Switch*). Эти элементы аналогичны переменным типа Boolean в языках высокого уровня и имеют два значения – *True* (1) или *False* (0). На панели *Block Diagram* эти элементы имеют рамку зеленого цвета, а соединительные провода – зеленая пунктирная линия. Из окна *(Controls/Express/Num Ind*) перетащите числовой индикатор *Numeric*, чтобы использовать его в качестве индикатора принятого сигнала.

Для того чтобы виртуальный инструмент работал в непрерывном режиме, в окне *Block Diagram* нужно выделить все элементы в рамку цикла *While Loop*. После установки этих элементов вид лицевой панели показан на рис.3.12, а блок-схемы – на рис.3.13. Обратите внимание на порядок расположения тумблеров!

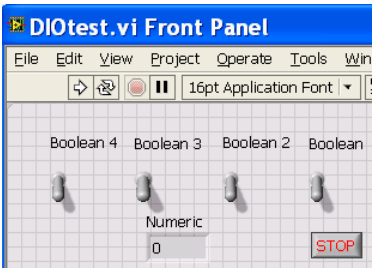

Рис.3.12. Вид рабочей области лицевой панели виртуального прибора.

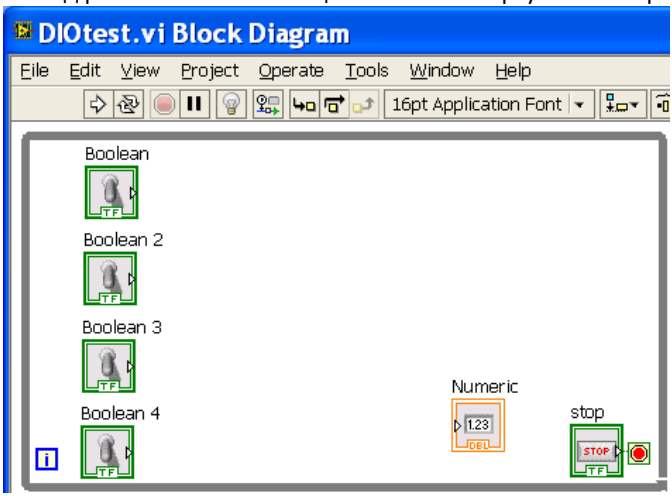

Рис.3.13. Вид блок-схемы после установки тумблеров и числового индикатора.

Для создания выходных цифровых электрических сигналов необходимо воспользоваться внешним устройством NI myDAQ. На его вход с компьютера поступают программно сформированные данные (двоичные коды), а на выходных контактах будут появляться соответствующие этим кодам реальные (а не виртуальные!) сигналы. При записи в порт электрические сигналы формируются на выходе цифровых микросхем транзисторно-транзисторной логики (ТТЛ), а при записи в ЦАП – формируется аналоговый сигнал на выходе ЦАП.

Для этого в окне *Block Diagram* выберите элемент *DAQ Assist* (*Functions/Express/Output/DAQ Assist)* и перетащите его иконку в окно *Block Diagram*. Появится диалоговое окно настройки *DAQ Assist* (рис.3.14), для создания порта вывода перейдите в этом окне в раздел *Generate Signals/Digital Output/Line Output* Generate и нажмите *Line Output.* Появится окно выбора порта цифрового выхода (рис.3.15), в нем выберите выходы *port0/line 4, port0/line 5, port0/line 6, port0/line 7*, которые будут задействованы для выходного сигнала. Нажмите кнопку *Finish* – появится окно дополнительной настройки линий ввода-вывода (рис.3.16).

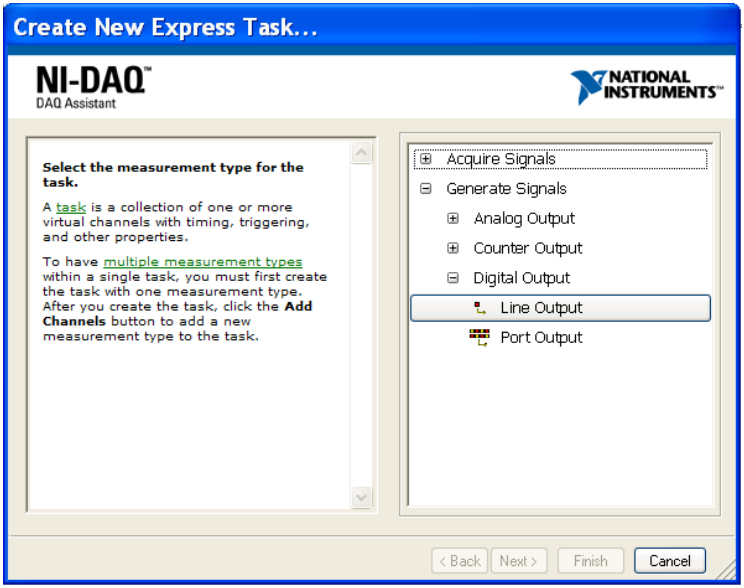

Рис.3.14. Выбор цифрового выхода в *DAQ Assistant*.

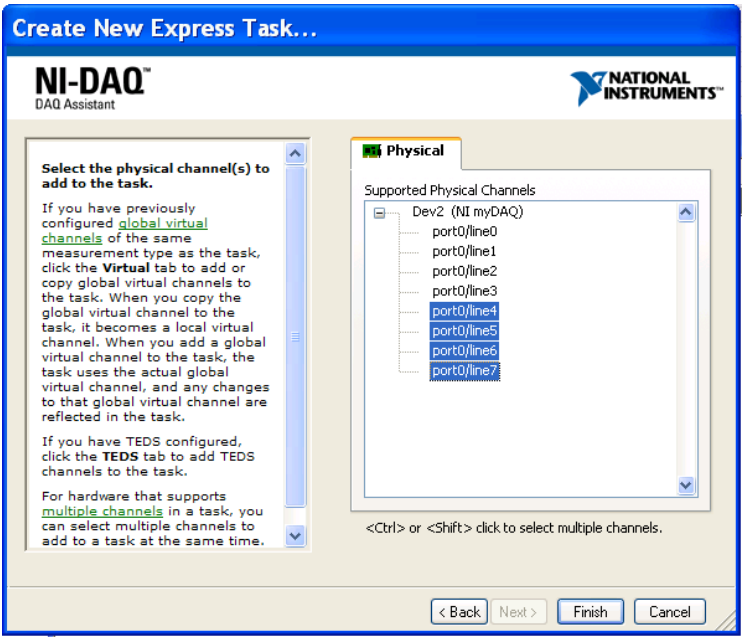

Рис.3.15. Выбор в *DAQ Assistant* линий порта для работы в режиме цифровых выходов.

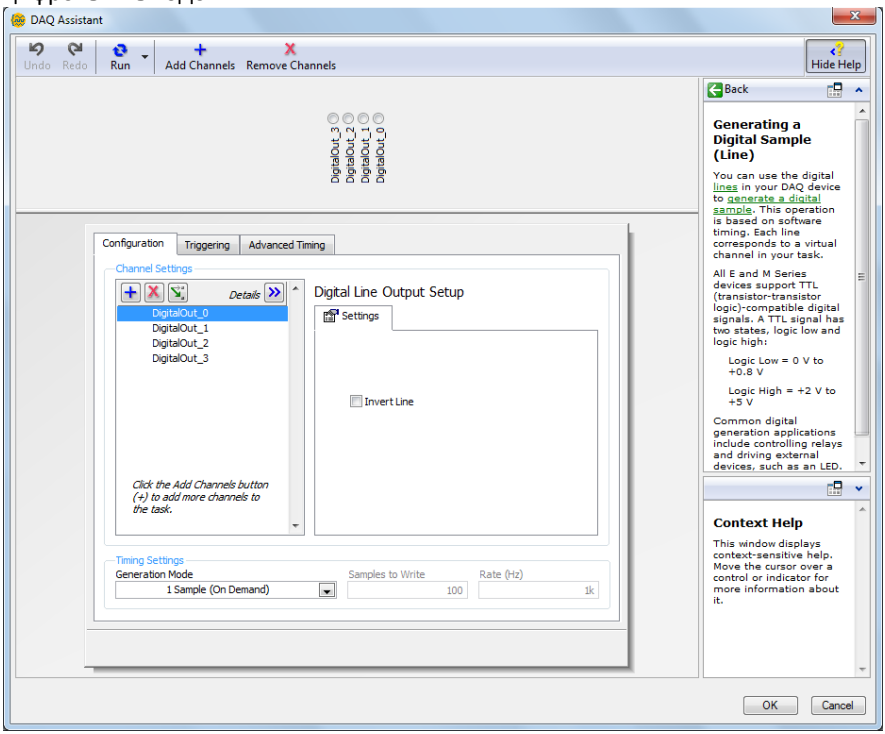

Рис.3.16. Дополнительные настройки в *DAQ Assistant* порта цифрового выхода.

Кнопки в разделе *Configuration* позволяют:  – просмотреть имеющуюся конфигурацию,  – добавить новый канал,  – удалить выбранные каналы,  – сменить конфигурацию выбранного канала.

Необходимо поставить точки напротив вертикальных надписей *DigitalOut_0 - DigitalOut_3* в верхней части экрана (хотя может сработать и без этого) и нажать кнопку *OK*. В результате линии порта *DIO 4, DIO 5, DIO 6, DIO 7* настроены в качестве **выходов**.

Сделанные настройки в любой момент можно изменить как вручную, аналогично проделанным ранее действиям (что будет сделано далее в рамках

данной лабораторной работы), так и программно с помощью специальных компонентов LabView.

На данном этапе созданная программа имеет вид, представленный на рис.3.17.

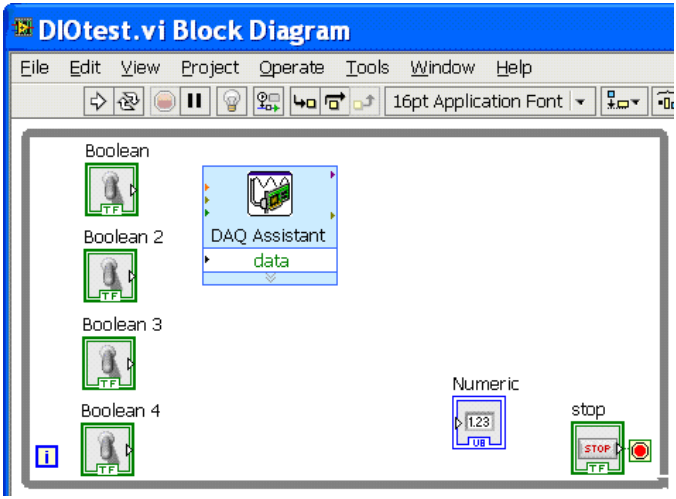

Рис.3.17. Вид программы после конфигурации порта вывода.

На следующем этапе необходимо представить данные, вводимые с помощью тумблеров, в виде массива чисел типа *Boolean*, с которыми может работать компонент *DAQ Assistant*. Для этого необходимо использовать функцию *Build Array,* расположенную в разделе *Functions/Programming/ Array/Build Array*. После добавления этой функции на блок-схему, нужно навести курсор мыши на этот компонент, чтобы появилась стрелка ↕, и растянуть его за верхнюю или нижнюю грань так, чтобы создать четыре входа для функции. Далее соедините входы *Build Array* с выходами тумблеров *Boolean*, а выход *Build Array* с входом *DAQ Assistant*.

Напоминаем, что *компонент DAQ Assistant, на вход которого поступают программные сигналы, передаваемые внутри программы LabVIEW, передает эти сигналы в цифровом виде с компьютера на устройство NI myDAQ, а оно преобразует их в электрические сигналы на своих выходах*. Если вывод

осуществляется в цифровой порт, на его клеммах появляются сигналы, соответствующие логическому нулю (допустимо любое значение в диапазоне от 0 до 0.4 В) или логической единице (любое значение в диапазоне от 2.4 В до 5.0 В). Если вывод осуществляется в ЦАП, на выходе ЦАП появляется напряжение, задаваемое поступившим двоичным кодом.

После произведенных операций должна получиться программа, изображенная на рис.3.18.

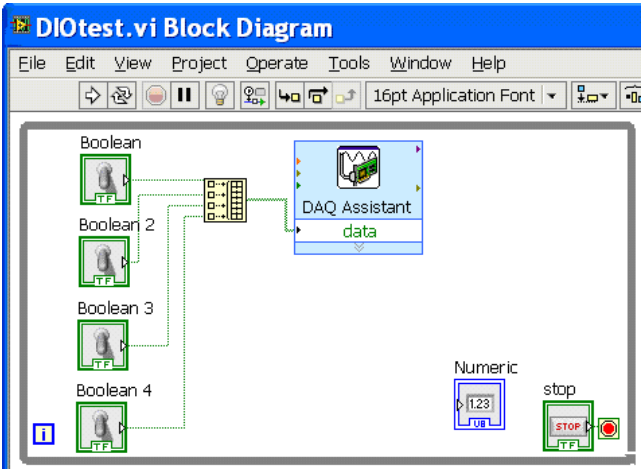

Рис.3.18. Добавление функции конвертации числа в массив и подача его на *DAQ Assistant*.

Мы хотим видеть на числовом индикаторе набранное тумблерами число, биты которого подаются на выходной порт NI myDAQ. Для этого необходимо преобразовать полученный массив в число. Проделайте это, воспользовавшись функцией *Boolean Array to Number*, расположенной в *Functions/Programming/ Numeric/Conversion*. После этого конечный код программы выглядит так, как представлено на рисунке 3.19:

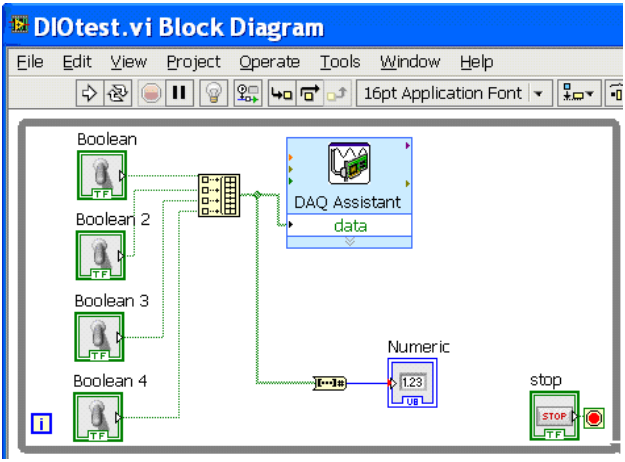

Рис.3.19. Добавление функции преобразования массива в число.

5.2.2. *Проверка работы программы – индикация числовых значений*

Запустите программу и убедитесь, что она работает. Проверьте, какие числа возникают на индикаторе *Numeric* при включении различных комбинаций тумблеров.

5.2.3. *Проверка работы программы – измерение сигналов на выходе порта*

Запустите виртуальный мультиметр в режиме вольтметра (можно также использовать обычный не виртуальный мультиметр) и с его помощью проверьте, как меняется напряжение на выходах порта ввода-вывода NI myDAQ – на клеммах *DIO 4, DIO 5, DIO 6, DIO 7.*

5.2.4. *Отображение на индикаторе двоичного кода, подаваемого на порт вывода*

Проведите настройку отображения числового индикатора так, чтобы он показывал двоичный код чисел. Для этого необходимо выполнить следующие действия: щелкните правой кнопкой мыши по значку индикатора и зайдите в раздел *Properties*, а затем во вкладку *Data Type.* В открывшемся окне нажмите на ярлык под *Representation*, выберите тип U8 (беззнаковое 8-битовое целое) – рис.3.20, нажмите *OK*.

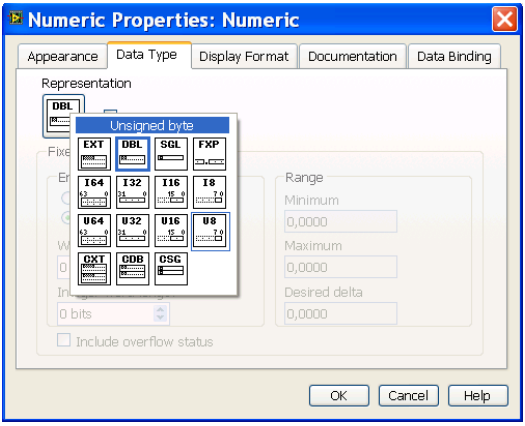

Рис.3.20. Установка типа показываемой индикатором *Numeric* величины.

Еще раз щелкните правой кнопкой мыши по значку индикатора и зайдите в раздел *Properties*. Во вкладке *Display Format* в поле *Type* выберите *Binary* (двоичный) и нажмите *OK* (рис.3.21).

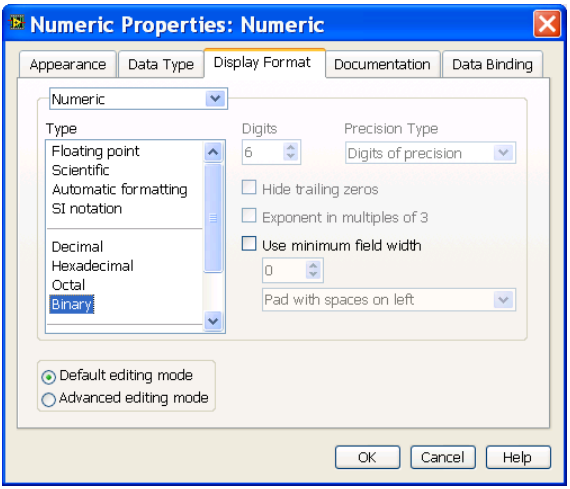

Рис.3.21. Установка формы представления показываемой индикатором *Numeric* величины.

Запустите программу. После этого при изменении положения тумблера должно изменяться и значение числового индикатора (0 или 1 в

соответствующем разряде). На рис.3.22 и рис.3.23 приведены примеры записи двоичных чисел с помощью тумблеров на *Front Panel.*

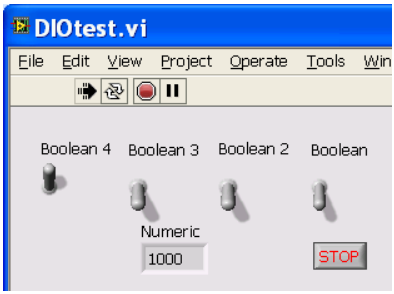

Рис.3.22. Логическая «1» подана на выход *DIO 7* (с *Boolean 4*).

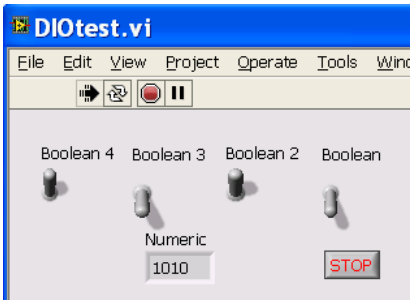

Рис.3.23 . Логическая «1» подана на выходы *DIO7* (с *Boolean 4)* и *DIO5* (с *Boolean 2*).

Поменяйте положение тумблеров, подавая с их помощью «0» или «1» на различные линии порта ввода/вывода.

Замечание: если в старших битах находятся нули, они не показываются. Для того чтобы в индикаторе показывались ведущие нули, необходимо зайти в свойства во вкладку *Display Format*, установить галочку "*Use minimum field width*" (использовать минимальную ширину поля вывода) – и далее значение 4 для ширины поля вывода. После чего в выпадающем списке необходимо выбрать режим вывода цифр "*Pad with zeros on left*" (дополняются нулями слева) и нажать *OK*.

Обратите внимание на то, какие тумблеры соответствуют старшим, а какие – младшим битам числа. В программировании принято нумеровать биты начиная с нуля (номер 0, 1, 2, 3 и т.д.). При этом бит с номером 0 считается младшим (самый правый в записи числа), и им кодируются числа 0_2 и 1_2. Булевское значение *False* кодируется в бите числом 0, а *True* – числом 1, про

эти значения говорят как про логический ноль и логическую единицу. Бит с номером 1 кодирует второй справа бит двоичного числа, и т.д.

При подаче с первого тумблера (имеющего имя *Boolean)* значения *True* (тумблер включен) в первую ячейку массива, т.е. имеющую индекс 0, будет записано значение “логическая 1”. Это значение будет передано на выход – на линию с наименьшим номером, т.е. на *DigitalOut_0* (см. рис. 3.16). В нашем случае *DigitalOut_0* соответствует линии 4 (*DIO 4*), поэтому электрический сигнал, соответствующий “логической 1”, попадет на клемму *DIO 4*. После преобразования массива в целое число значение элемента массива с индексом 0 соответствует младшему биту числа, поэтому на индикаторе покажется число 1_2.

Какое минимальное и какое максимальное десятичное число можно записать в порт с помощью созданной программы?

Задание 5.3. Отображение считываемых из порта цифровых сигналов в виде двоичных чисел

В этом задании нужно создать программу, которая считывает цифровые сигналы с линий *DIO 0, DIO 1, DIO 2, DIO 3*, которые в предыдущем задании не использовались – в данном задании они используются в качестве **входов**. Считанные с помощью NI myDAQ сигналы необходимо отобразить на числовом индикаторе в виде двоичного числа.

Общепринятая методика написания сложной программы – создавать и отлаживать независимые части программы, а затем соединять их в единое работающее целое. Поэтому ту часть программы, которая относилась к выводу данных из порта, и была написана и проверена в предыдущем задании, мы не меняем и в данном задании не используем. (Совместное использование этих двух частей будет проведено в задании 5.4).

Виртуальный инструмент для считывания цифровых сигналов с линий ввода настраивается аналогично тому, как это было сделано для линий вывода. Перетащите в окно *Block Diagram* еще один компонент *DAQ Assistant.* Его можно брать как из *Functions/Express/Output*, так и из раздела

Functions/Express/Input – это один и тот же компонент, а на вход или выход он работает в зависимости от сделанных настроек.

В диалоговом окне настройки, появившемся после перетаскивания компонента на блок-схему (рис.3.24), перейдите в *Line Input* (*Acquire Signals/Digital* ***Input****/Line Input*), и выберите необходимый порт и входы *DIO 0 – DIO 3 (port0/line 0, port0/line 1, port0/line 2, port0/line 3*) .

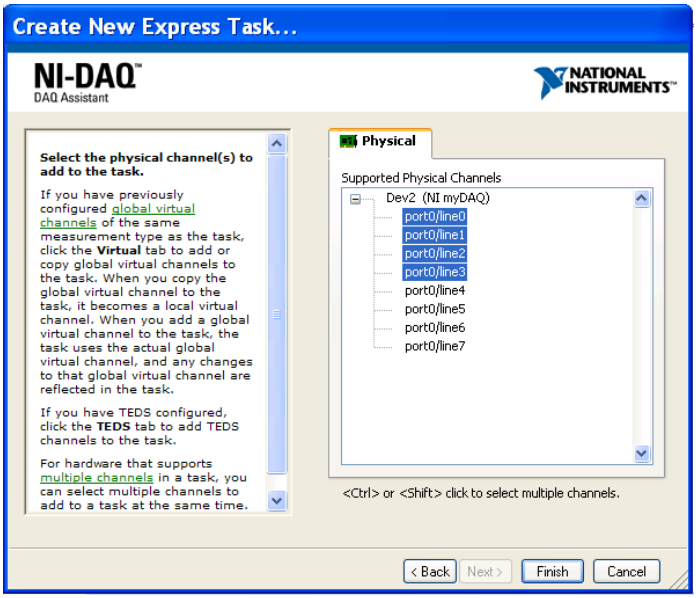

Рис.3.24. Выбор линий порта для работы в режиме цифровых входов.

После этого нажмите кнопку *Finish*, и в появившемся окне конфигурации линий нажмите кнопку *OK*. В результате линии *DIO 0, DIO 1, DIO 2, DIO 3* окажутся настроены в качестве цифровых **входов** порта ввода-вывода.

На данном этапе созданная программа имеет вид, представленный на рис.3.25.

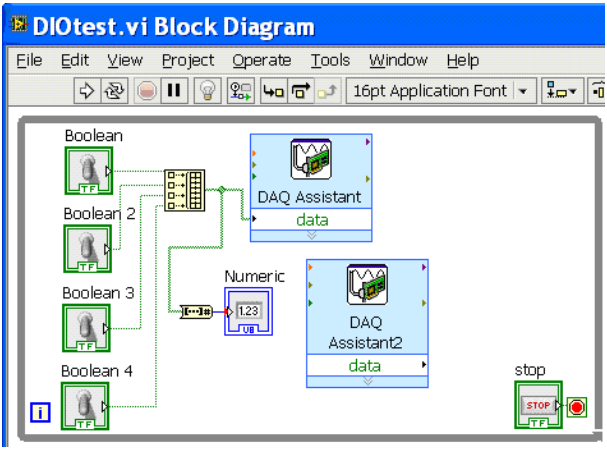

Рис.3.25. Вид программы после конфигурации порта ввода.

Входной сигнал необходимо преобразовать из массива в число. Проделайте это, воспользовавшись функцией *Boolean Array to Number*, расположенной в *Functions/Programming/ Numeric/Conversion*. После этого добавьте второй числовой индикатор для показа полученного числа и настройте его для показа двоичного кода. Конечный код программы показан на рис.3.26. Можете параллельно с индикатором *Numeric2* подсоединить еще один, показывающий считанное число в десятичном виде.

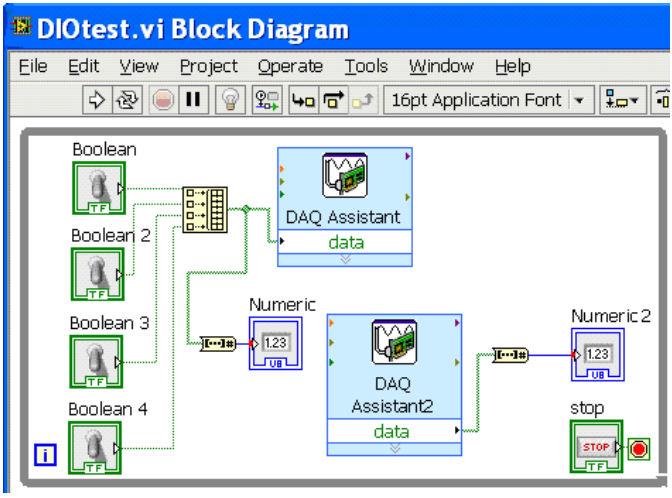

Рис.3.26. Программа с индикацией считанного из порта числа.

Запустите программу. Какое считанное значение показывается?

На наборной панели соедините клемму *DIO 0* через резистор с номиналом от 100 Ом до 3 Ком проводами (перемычками)к клемме с напряжением +5 В. Какое считанное значение показывается? Что будет, если подсоединить клемму не к +5 В, а к клемме “земля”?

Проведите аналогичные эксперименты с клеммами *DIO 1, DIO 2. DIO 3.*

Какую комбинацию сигналов надо подать на входы, чтобы получить на индикаторе код 1011? Какому десятичному числу соответствует этот код?

Сохраните созданный вами проект.

Замечание: Прибор NI MyDAQ после программирования портов в дальнейшем сохраняет запрограммированное состояние даже после отключения питания. Поэтому при выполнении задания, в котором используется копия предыдущего проекта, для правильной работы нового проекта требуется снова перепрограммировать порты в диалоговом окне настройки компонента DAQ Assistant в соответствии с рис.3.24.

Задание 5.4. Одновременная работа с записью цифровых данных в порт вывода и считыванием цифровых данных из порта ввода

В этом задании нужно использовать разработанную программу для того, чтобы отправить сформированный с помощью виртуальных тумблеров цифровой сигнал на линии вывода *DIO 4, DIO 5, DIO 6, DIO 7* (они используются в качестве **выходов**), а затем этот электрический сигнал считать с помощью линий *DIO 0, DIO 1, DIO 2, DIO 3* – в данном задании они используются в качестве **входов**. Эти сигналы будут отображены числовым индикатором *Numeric2* в виде двоичного числа.

5.4.1*. Соединение одного аппаратного цифрового выхода с одним аппаратным цифровым входом*

На наборной панели соедините клемму *DIO 4* (выход) с клеммой *DIO 0* (вход). Какой тумблер надо включить для того, чтобы на индикаторе

Numeric2 появилось число, отличное от нуля? Какое число появляется? Почему?

Затем соедините клемму *DIO 4* с клеммой *DIO 1* (вход). Какой тумблер надо включить для того, чтобы на индикаторе *Numeric2* появилось число 1?

С каким входом необходимо соединить клемму *DIO 4* (выход), чтобы при включении тумблера, с которого на нее подается единица, на индикаторе *Numeric2* появилось 10_2?

Отключите клемму *DIO 4* от входных линий порта NI myDAQ. Соедините клемму *DIO 5* (выход) с клеммой *DIO 1* (вход). Какой тумблер надо включить для того, чтобы на индикаторе *Numeric2* появилось число, отличное от нуля? Какое число появляется? Почему?

5.4.2. *Соединение одного аппаратного цифрового выхода с несколькими аппаратными цифровыми входами*

На наборной панели соедините клемму *DIO 5* (выход) кроме клеммы *DIO 1* еще с одним входом – с клеммой *DIO 0.*

Какой двоичный код подается на выходные линии порта при наборе тумблерами кода 1000_2? Какой код при этом подается на входные линии порта?

Какой двоичный код подается на выходные линии порта при наборе тумблерами кода 0100_2? Какой код при этом подается на входные линии порта?

Какой двоичный код подается на выходные линии порта при наборе тумблерами кода 0110_2? Какой код при этом подается на входные линии порта? Проверьте ваше утверждение измерением напряжений на клеммах с помощью мультиметра.

5.4.3. *Соединение всех аппаратных цифровых выходов с аппаратными цифровыми входами*

На наборной панели соедините проводами (перемычками) выход *DIO 4* и вход *DIO 0,* выход *DIO 5* и вход *DIO 1*, выход *DIO 6* и вход *DIO 2,* выход *DIO 7* и вход *DIO 3*. Проверьте, что благодаря такому соединению числа, которые

будут записаны на выходы порта *DIO4–DIO7*, превратятся в электрические сигналы, поданные с выходных контактов на входы порта *DIO0–DIO3*, которые программа прочитает и отобразит в содержимом компонента *Numeric 2*.

Что произойдет, если на наборной панели вынуть перемычку для линии *DIO 7* и переключать виртуальные тумблеры? Если на наборной панели вынуть перемычку для линии *DIO 4*? Если сигнал с *DIO 7* подать на *DIO 0*, а с *DIO 4* подать на *DIO 3*?

Задание 5.5. Отображение цифровых сигналов на виртуальных светодиодах

Сделайте работу вашей программы более наглядной с использованием виртуальных светодиодов (рис.3.27– 3.28).

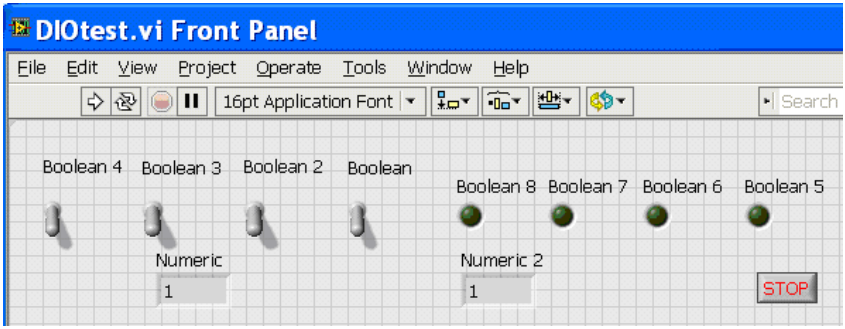

Рис.3.27. Добавление виртуальных светодиодов на *Front Panel*.

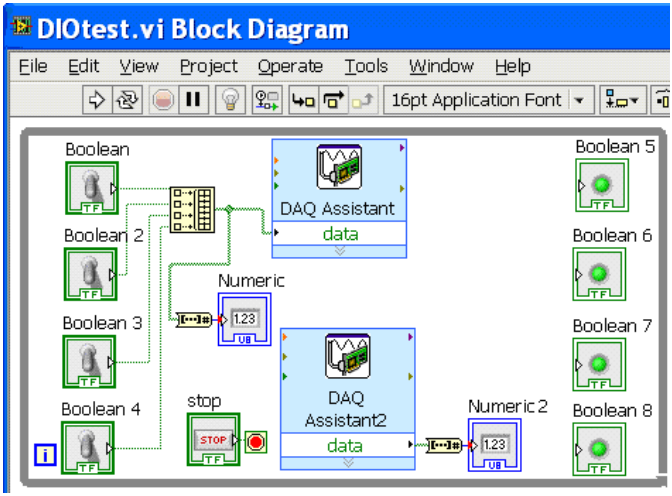

Рис.3.28. Добавление виртуальных светодиодов в *Block Diagram*.

На выходе *DAQ Assistant* в результате измерения сигнала с порта ввода имеется булевский массив. Но точно так же, как его приходилось сначала преобразовывать для вывода на индикатор *Numeric*, и в данном случае необходимо выполнить преобразования. Сначала требуется преобразовать этот массив к кластеру из булевых переменных с помощью компонента *Array To Cluster*, расположенного в разделе *Functions/Programming/Array* (рис.3.29).

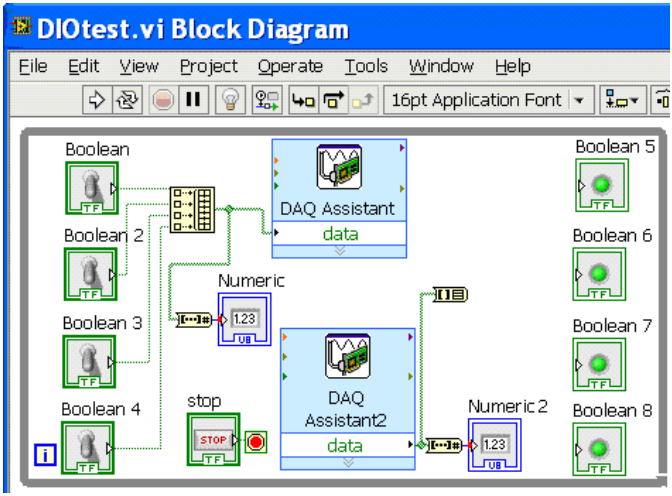

Рис.3.29. Добавление инструмента *Array To Cluster*

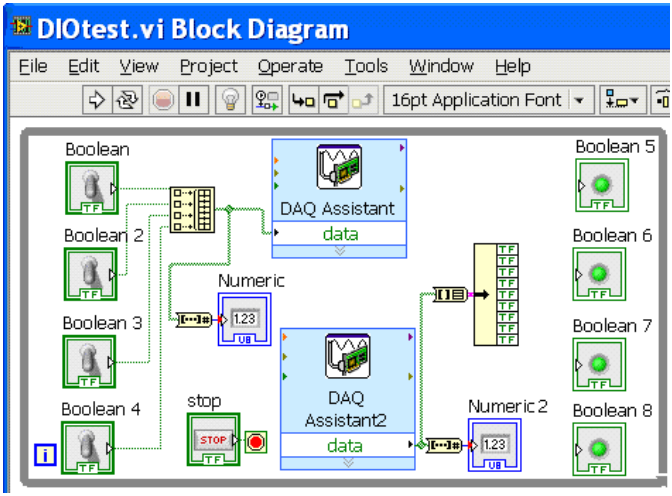

Рис.3.30. Добавление инструмента *Unbundle.*

Далее получившийся кластер можно разбить на отдельные биты инструментом *Unbundle*, расположенным в разделе *Functions/Programming/ Cluster,Class&Variant.* После соединения выхода *Array To Cluster* с инструментом *Unbundle* автоматически определится количество элементов в кластере (рис.3.30).

Учитывая, что верхний выход инструмента *Unbundle* отвечает за *младший* бит входного сигнала (с клеммы *DIO 0*), а нижний – за старший (с клеммы *DIO 3*), следует соединить их с соответствующими входами виртуальных диодов так, как показано на рис.3.31. Пример работы созданной программы показан на рис.3.32.

Сохраните созданный проект.

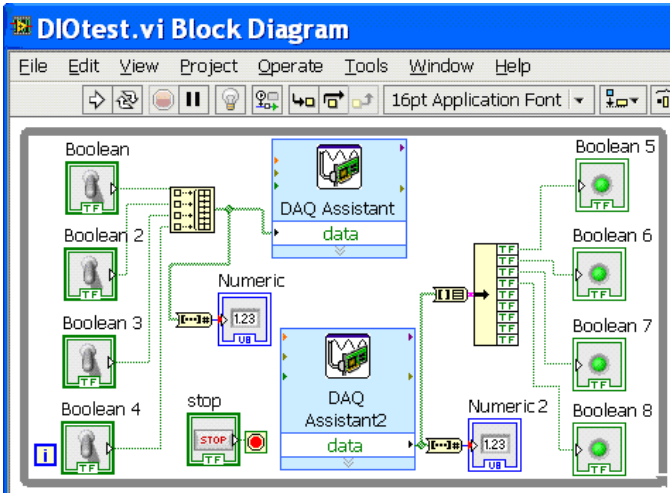

Рис.3.31. Итоговый вид программы.

Рис.3.32 . Тестирование программы с виртуальными светодиодами.

Потренируйтесь в записи различных чисел в порт с помощью тумблеров. Какое максимальное десятичное число можно записать в порт с помощью созданной программы без потери информации?

Какое десятичное число нужно записать в порт, чтобы светодиоды горели через один – четные диоды? Нечетные диоды?

Что произойдет, если сигнал с выхода *DIO 4* подать не на вход *DIO 0*, а на вход *DIO 1*, и при этом сигнал с выхода *DIO 5* подать не на вход *DIO 1*, а на вход *DIO 0*?

Что произойдет, если на наборной панели соединить проводами (перемычками) клеммы *DIO 7* и *DIO 0*, *DIO 6* и *DIO 1*, *DIO 5* и *DIO 2*, *DIO 4* и *DIO 3*?

Задание 5.6. Отображение цифровых сигналов с помощью реальных светодиодов: все линии порта в режиме вывода, запись в порт десятичных чисел, вводимых с клавиатуры

5.6.1. Электрическая схема установки

В этом задании необходимо научиться записывать в порт десятичные числа, вводимые в пункт ввода *Numeric* с помощью клавиатуры, запрограммировав на выход восемь цифровых линий *DIO 0 – DIO 7*. Для индикации сигнала на каждой линии используйте восемь светодиодов, подсоединенных к цифровой земле через резисторы.

Соберите на наборной панели электрическую схему, приведенную на рис.3.33.

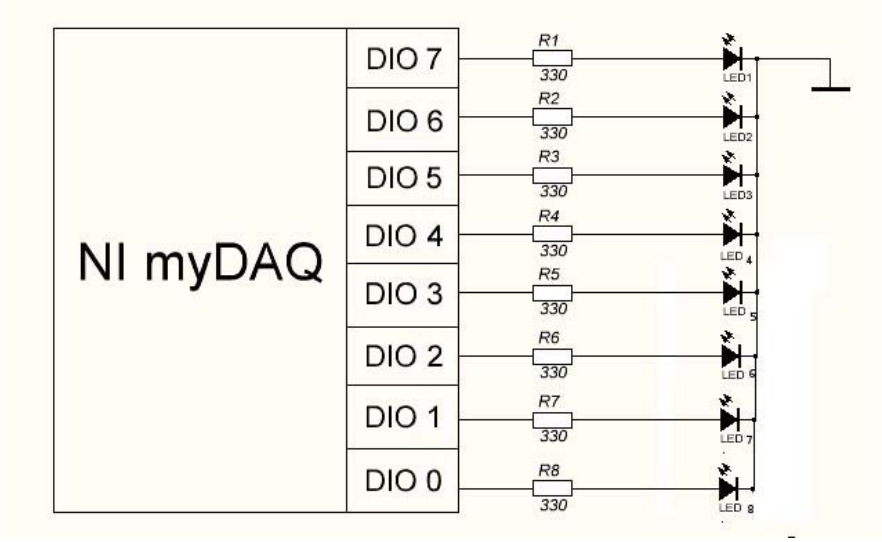

Рис. 3.33. Электрическая схема собираемой установки.

5.6.2. Начало написания программы для вывода данных в порт NI myDAQ

На лицевую панель виртуального прибора для отображения вводимого с клавиатуры числа поместите компонент *Numeric*.

В окне *Block Diagram* поместите основной блок программы в конструкцию *While Loop* с кнопкой *Stop*, чтобы виртуальный инструмент работал в непрерывном режиме.

Для вывода данных в порт NI myDAQ сначала необходимо поместить на блок-схему компонент *DAQ Assistant.*

5.6.3. Перепрограммирование 8 линий порта NI myDAQ на вывод электрических сигналов

В открывшемся окне для создания порта вывода необходимо выбрать раздел *Generate Signals/Digital Output/Line Output*, выбрать *Port Output* (рис.3.15), и, аналогично тому, как показано на рис.3.16, удерживая клавишу Shift, выбрать все клеммы вывода, от *port0/line 0* до *port0/line 7.* Затем следует нажать кнопку *Finish*.

В появившемся окне (рис.3.34) необходимо поставить точки напротив вертикальных надписей *DigitalOut_0* - *DigitalOut_7* в верхней части экрана (хотя может сработать и без этого) и нажать кнопку *OK*.

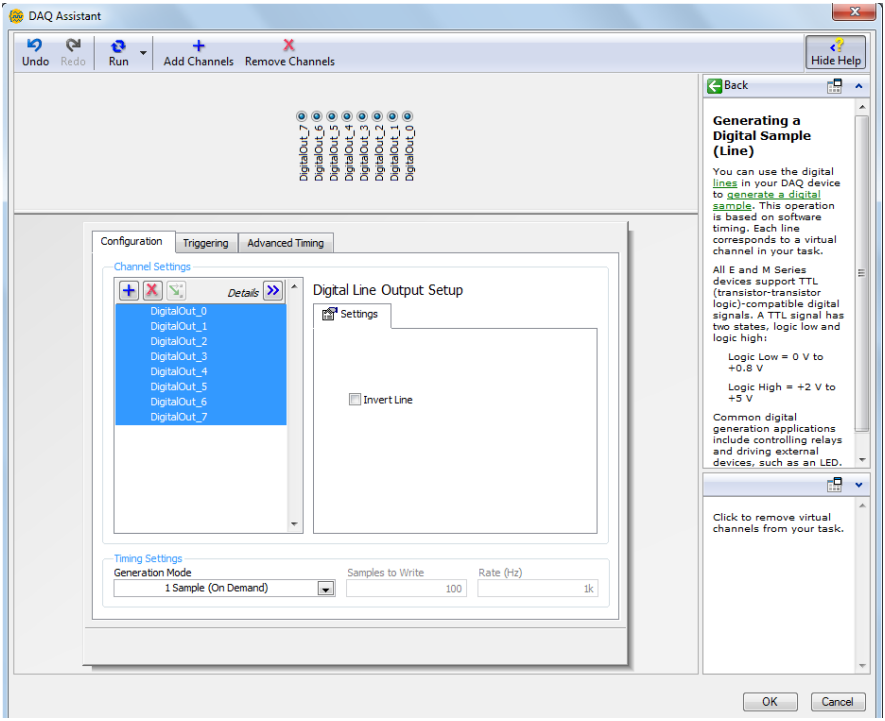

Рис.3.34. Дополнительные настройки портов цифрового выхода.

После этого в окне блок-схемы добавится иконка *DAQ Assistant*.

5.6.4. Сопоставление компоненту Numeric числового типа U8

Для корректной передачи данных на порт вывода компоненту *Numeric* необходимо сопоставить числовой тип U8 (беззнаковое 8-битное целое). Раньше мы подавали сигнал из меньшего чем 8 числа бит, и подобной проблемы не возникало. Для сопоставления компоненту *Numeric* данного числового типа следует щелкнуть правой кнопкой мыши на иконке *Numeric*, и выбрать пункт *Representation/U8* (рис.3.35) – либо пункт *Properties* и вкладку *Data Type*.

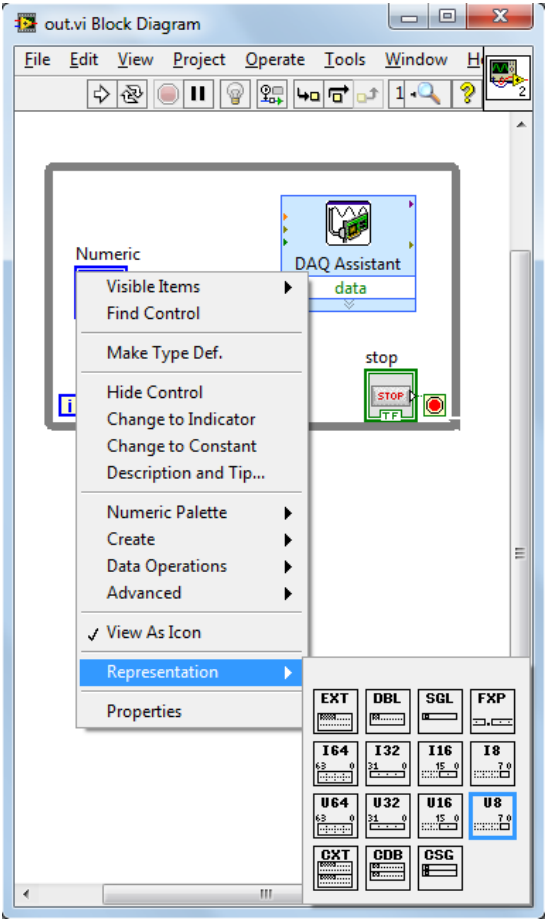

Рис.3.35. Задание числового типа, сопоставляемого данным компонента *Numeric*.

5.6.5. Преобразование и обработка введенных с клавиатуры данных

Далее необходимо представить данные, вводимые с клавиатуры в пункт *Numeric*, в виде массива чисел типа *Boolean*. Для этого необходимо воспользоваться функцией *Number* to *Boolean Array,* расположенной в разделе *Functions/Programming/Numeric/Conversion.* В результате добавления этой функции получается программа, изображенная на рис.3.36.

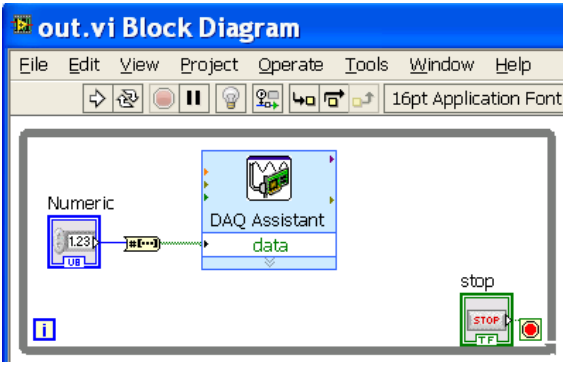

Рис.3.36. Промежуточный вид функциональной части программы.

Для удобства работы на лицевую панель добавьте кнопку управления *Push Button* (из раздела *Controls/Express/Buttons&Switches*), при включении которой вводимое число в двоичном представлении будет отправляться на выход NI myDAQ (рис.3.37 и рис.3.38).

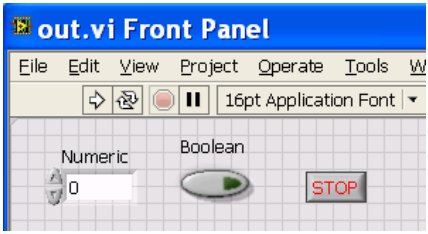

Рис.3.37. Вид визуальной части программы после добавления кнопки управления.

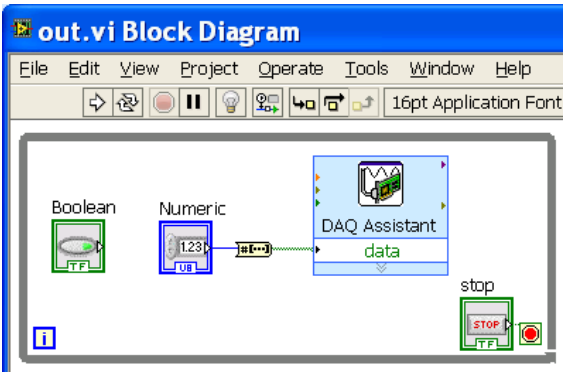

Рис.3.38. Вид программной части после добавления кнопки управления.

Push Button – это кнопка, которая при нажатии фиксируется в нажатом состоянии – включается. Повторное нажатие выключает кнопку – возвращает ее в не нажатое состояние.

Для использования этой кнопки на блок-схему необходимо добавить компонент *Case Structure (Functions/Programming/Structures/Case Structure).* Далее необходимо переместить компонент ввода *Numeric* внутрь рамки компонента *Case Structure*, а кнопку *Boolean* подключить к зеленому знаку вопроса компонента *Case Structure* (рис.3.39).

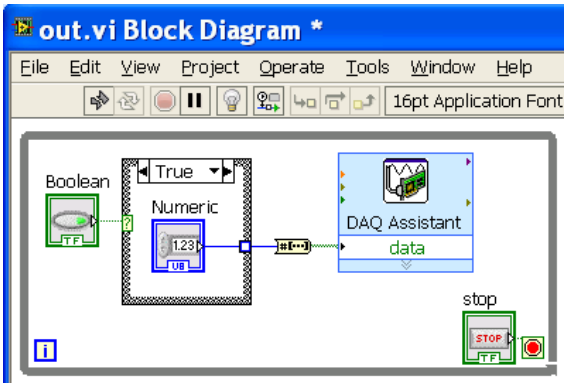

Рис.3.39. Добавление компонента *Case Structure*.

По умолчанию видна вкладка True компонента *Case Structure,* и при поступлении на вход, помеченный знаком вопроса, значения True, на выход компонента будут поступать данные с компонента *Numeric*.

Далее в верхней части компонента *Case Structure* необходимо поменять вкладку *True на False* для ситуации, когда на вход компонента поступает значение *False.* После смены вкладки с *True на False* первоначально внутри рамки структуры никаких элементов нет – туда необходимо добавить числовую константу 0 и соединить ее с квадратиком, расположенным на правой стороне конструкции *Case Structure* (рис.3.40), который после подсоединения становится полностью заполненным синим цветом.

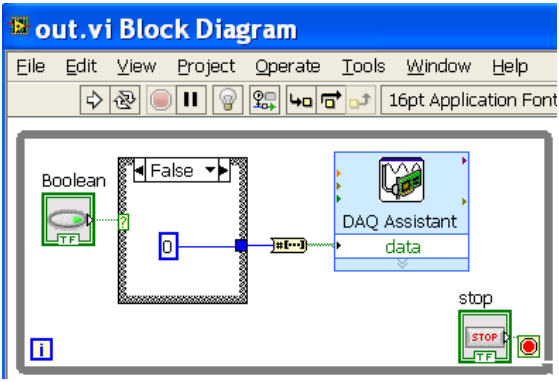

Рис.3.40. Режим False в компоненте *Case Structure*.

Для корректной работы программы в настройках необходимо задать в качестве типа константы U8 – в противном случае программа запустится, но при ее работе возникнет ошибка: при выключенной кнопке *Boolean* на вход *DAQ Assistant* подается массив из 32 булевских значений, а линий порта всего 8.

5.6.6. Запуск программы и контрольные вопросы

Запустите программу. Проверьте, как зажигаются светодиоды при вводе различных чисел.

- Какое число надо ввести, чтобы зажечь все светодиоды?
- Чтобы зажечь светодиоды с нечётными номерами (нумерация идёт от нуля)?
- Чтобы зажечь светодиоды с чётными номерами?
- Что произойдёт, если ввести в пункт ввода число 100? Число 256? Число 1000? Почему?
- Что произойдёт, если сменить тип данных в компоненте *Numeric* на U16? Почему?

Задание 5.7. Чтение цифровых сигналов с 8 входов порта и их отображение на 8 виртуальных светодиодах

NI myDAQ надо запрограммировать так, чтобы линии порта *DIO 0 – DIO 7* использовались в качестве **входов**, и на них необходимо подавать логические

уровни напряжения снаружи. При этом напряжение +5В считается уровнем логической единицы, а напряжение, подаваемое с “земли” (0 В) – уровнем логического нуля.

По схеме, представленной на рис.3.41, соберите на наборной панели установку, состоящую из подключенных к портам вывода резисторов 330 Ом. «Напряжение +5В» сформируйте с помощью перемычек.

Ко **входным** клеммам *DIO 0 – DIO 7* порта подсоедините ключи (тумблеры) *S1– S7*, через резисторы подключенные к +5В. При отсутствии ключей используйте перемычки, с помощью которых можно соединять клеммы наборной панели.

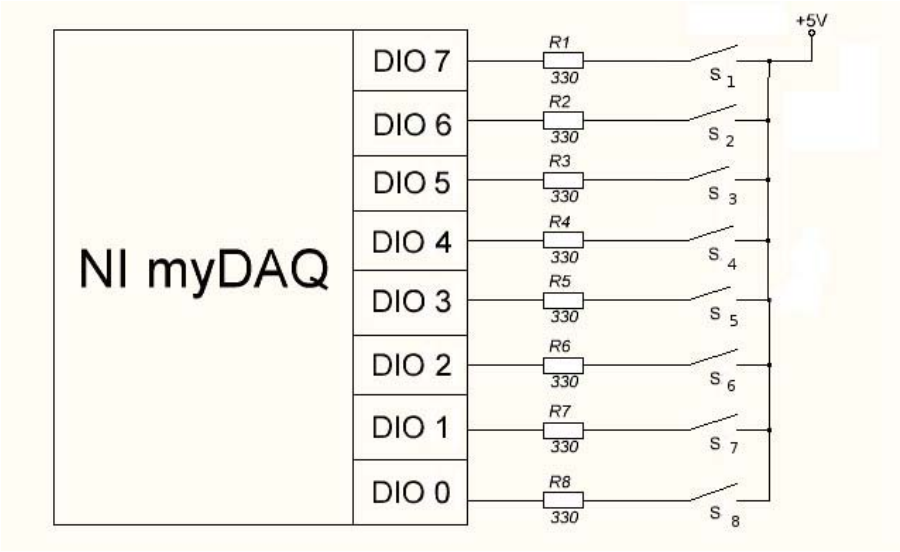

Рис 3.41. Электрическая схема собираемой установки.

Замечание 1: при разомкнутых ключах *S1 – S7* входы *DIO 0 – DIO 7* оказываются “висящими” – на них не подаётся ни логический 0, ни логическая 1. Однако внутри NI myDAQ эти клеммы через специальные резисторы , называемые подтягивающими вниз (*pull-down*), подсоединяются к цифровой земле, поэтому при отсутствии внешнего цифрового сигнала на эти входы подаётся логический 0.

Замечание 2: это удачная особенность NI myDAQ – чаще электронные устройства оснащаются внутренними подтягивающими вверх (*pull-up*)

резисторами, и при отсутствии внешнего сигнала на входах имеется уровень логической единицы. Что гораздо менее удобно.

По аналогии с заданием 5.5 напишите программу, отображающую сигналы, читаемые из линий порта, на 8 виртуальных светодиодах. Сохраните ее в файле.

Задание 5.8. Отображение цифровых сигналов с помощью реальных светодиодов: 4 линии порта в режиме вывода, 4 линии порта в режиме ввода

1) Подумайте, какой вид должна иметь программа для отображения на 4 реальных светодиодах цифровых сигналов, считываемых с 4 входов *DIO 0 – DIO 3*, и почему электрическая схема, показанная на рис.3.42, позволит проводить такое отображение.

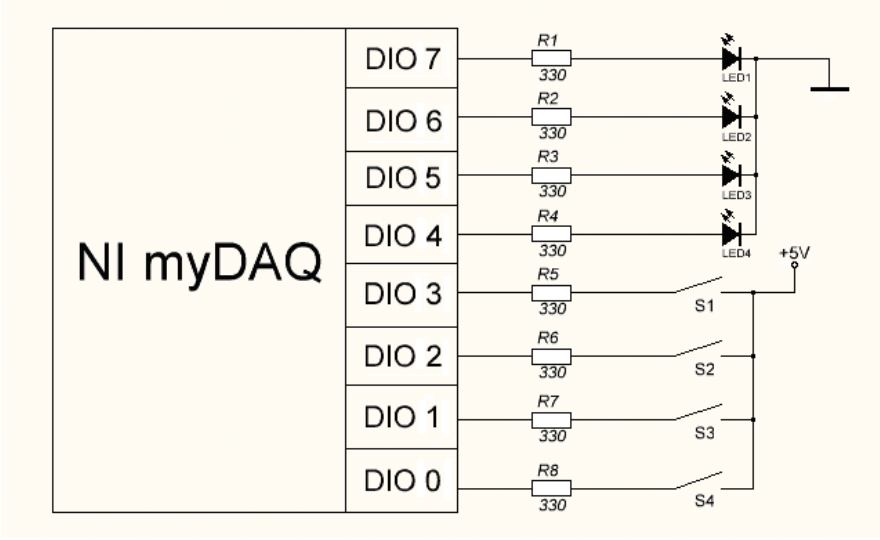

Рис.3.42. Электрическая схема собираемой установки.

NI myDAQ надо запрограммировать так (рис.3.15, рис.3.24), чтобы линии порта *DIO 0, DIO 1, DIO 2, DIO 3* использовались в качестве **входов**, и на них необходимо подавать логические уровни напряжения снаружи. А линии *DIO 4, DIO 5, DIO 6, DIO 7* надо запрограммировать для использования в качестве **выходов** – на них будут поступать уровни напряжения, вырабатываемые NI myDAQ.

Напишите программу для отображения цифровых сигналов на светодиодах, сохраните ее в файле.

2) По схеме, представленной на рис.3.42, соберите на наборной панели установку, состоящую из подключенных к портам вывода светодиодов (через резисторы 330 Ом). Шины « земля» и «напряжение +5В» сформируйте с помощью перемычек.

Ко **входным** клеммам *DIO 0 – DIO 3* порта подсоедините ключи (тумблеры) *S1– S4*, через резисторы подключенные к +5В. При отсутствии ключей используйте перемычки, с помощью которых можно соединять клеммы наборной панели.

3) Измерьте напряжение на **выходных** клеммах *DIO 4 – DIO 7* с помощью цифрового мультиметра DMM. На программной лицевой панели *NI ELVISmx* щелкните иконку прибора DMM, на виртуальной панели DMM нажмите кнопку V±, в поле *Range* выберите диапазон 20 В. На наборной панели подключите выходные разъемы мультиметра к соответствующим клеммам *DIO 4 – DIO 7*.

4) Осуществите вывод данных с наборной панели.

Зажгите:

- все светодиоды на наборной панели;
- четные светодиоды;
- нечетные светодиоды.

Контрольный вопрос: зачем нужно использовать два компонента *DAQ Assistant* (один для считывания сигналов с входов порта, другой – для записи в выходные линии порта), если можно сразу подать входные сигналы на реальные светодиоды? Для чего может потребоваться компьютер? Для чего при наличии компьютера может потребоваться индикация сигналов с помощью светодиодов?

Обратите внимание на то, что если у компонента *DAQ Assistant* имеется *Выход* программных данных – этот компонент *считывает* данные с аппаратных входов, а если у *DAQ Assistant* имеется *Вход*

программных данных – этот компонент *выводит* эти данные на аппаратные выходы.

Задание 5.9*. Отображение цифровых сигналов на реальных светодиодах с программной инверсией сигналов

Инверсия цифровых сигналов часто бывает необходима при управлении различными устройствами и при выводе сигналов на индикаторы. В данном задании изучается возможность программного управления инверсией сигналов.

Создайте копию программы из предыдущего задания. Внесите такое изменение в программу (без изменения электрической схемы, изображенной на рис.3.42), чтобы светодиоды зажигались противоположным образом по сравнению с предыдущим заданием: чтобы при замыкании тумблера соответствующий ему светодиод гас, а при размыкании – зажигался.

Подсказка: для всех бит на выходе *DAQ Assistant*, т.е. для линий *DigitalOut_0 – DigitalOut_3,* надо сделать программную инверсию сигнала. Проще всего это сделать в свойствах компонента *DAQ Assistant* (рис.3.16) – выбрать эти линии, поставить галочку *Invert Line* и нажать *OK.*

Посмотрите, какое напряжение будет появляться на выходных портах NI My DAQ после подачи на входные линии схемы сигналов. Убедитесь, что электрические сигналы инвертированы по сравнению со входными.

Задание 5.10*. Отображение цифровых сигналов на реальных светодиодах с аппаратной инверсией сигналов

Создайте копию программы из предыдущего задания и верните режим без инверсии выходного сигнала. Внесите такое изменение в электрической схеме, изображенной на рис.3.42, чтобы светодиоды подключались не к земле, а к +5 В. Убедитесь, что при этом они будут зажигаться так же, как и в предыдущем задании: что при замыкании выключателя соответствующий ему светодиод гаснет, а при размыкании – зажигается.

На практике часто бывает невозможно или затруднительно изменять электрическую схему. Представьте, что вам необходимо использовать

данную электрическую схему без внесения в нее изменений, но требуется, чтобы светодиоды загорались без инверсии: чтобы при замыкании выключателя соответствующий ему светодиод зажигался, а при размыкании – гас. Каким образом этого добиться? Реализуйте вашу идею и продемонстрируйте работу установки.

Задание 5.11*. Программа проверки равенства двоичных и десятичных чисел

Напишите программу, которая проверяет равенство двоичного числа, введенного пользователем в первый пункт ввода *Numeric,* и десятичного числа, введенного пользователем во второй пункт ввода *Numeric.*

В.В. Монахов, О.В. Огинец, С.Н. Жоголь, М.Г.Яковлева

Создание виртуальных приборов и программирование устройства сбора данных NI myDAQ в среде LABVIEW

Учебное пособие
2-е издание, переработанное и дополненное

Издание прошло редакционно-корректорскую правку

Подписано в печать 27.12.2017 г.
Формат 60x84 1/16. Бумага офсетная. Печать цифровая.
Усл. печ. л. 7,7. Тираж 50 экз.
Заказ № 4709
Отпечатано с готового оригинал-макета заказчика
в ООО «Издательство “ЛЕМА”»
199004, Россия, Санкт-Петербург, 1-я линия В.О., д.28
тел.: 323-30-50, тел./факс: 323-67-74
e-mail: izd_lema@mail.ru
http://www.lemaprint.ru