Уроки по LabVIEW

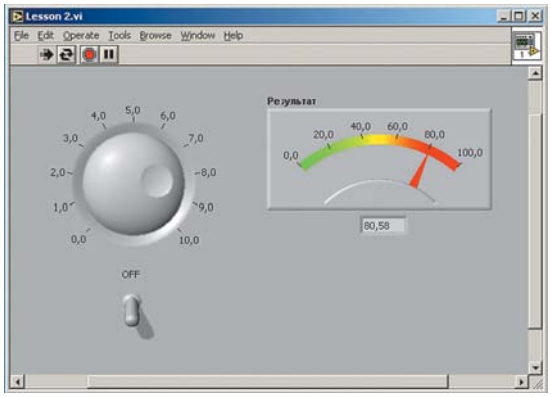
На этом уроке Вы научитесь:

- *использовать CASE-структуры в создаваемых приложениях*
- *применять циклы типа For-Do в своих программах*
- *изменять свойства графического элемента управления "Кнопка"*
- *обрабатывать нажатия кнопок*
- *использовать редактор формул для написания кода программы*
- *разрабатывать подпрограммы и сохранять их в виде отдельного виртуального инструмента*
- *оформлять иконку полученной подпрограммы и устанавливать связи иконки с используемыми в подпрограмме параметрами*
- *изменять свойства подпрограммы и сохранять их в виде отдельных файлов или в составе библиотеки*
- *использовать созданную подпрограмму в других приложениях*

Реализация даже элементарных алгоритмов, как правило, не обходится без операций логического ветвления программы в зависимости от определенных условий. Для этих целей используются так называемые ***Case***-структуры. Такие структуры позволяют осуществлять выбор по условию или по значению параметра-селектора и переходить на выполнение соответствующих действий.

Создадим элементарную программу, которая позволит, в зависимости от положения тумблера, получать различный результат вычислений. Если тумблер находится в выключенном состоянии, то значение входного параметра будет умножаться на 10, а в противном случае - меняться не будет.

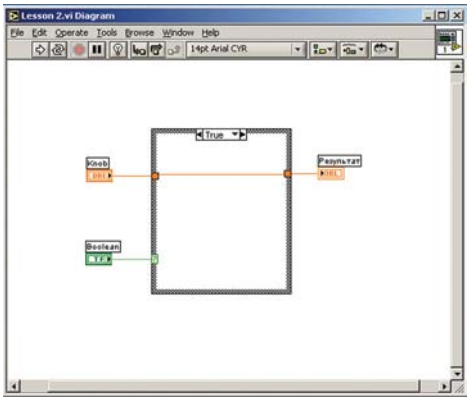
Приступим к реализации задачи. Создаем новый ***VI***. Формируем на панели графический интерфейс:



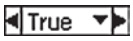
Элементы графического интерфейса, определяющие входные параметры размещаем слева, а выходные, т.е. результат - справа.

Теперь реализуем условный алгоритм. Переходим в окно редактирования диаграмм. Устанавливаем в это окно ***CASE***-структуру, которая находится в ***Functions*** >> ***Structures*** >> ***Case***. Изменяем ее размеры. Заметим, что по умолчанию состояние определено как ***True*** (истина).

Соединяем элементы диаграммы как показано на рисунке:

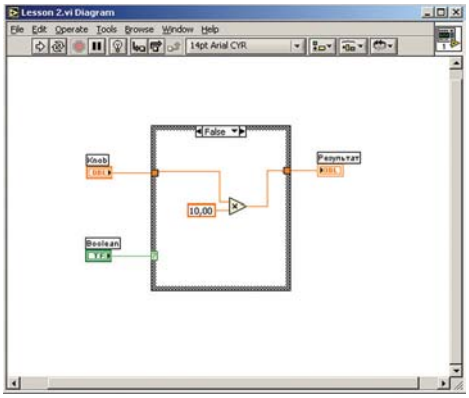


Переключаем режим ***CASE***-структуры на ***False*** (ложь). Для этого необходимо подвести указатель мыши к



одной из стрелок структуры (левая/правая) и нажать левую кнопку. Состояние изменится.

Согласно заданию реализуем алгоритм умножения значения входного параметра на 10. Для этого необходимо установить необходимые компоненты в окно редактирования диаграмм и соединить их:

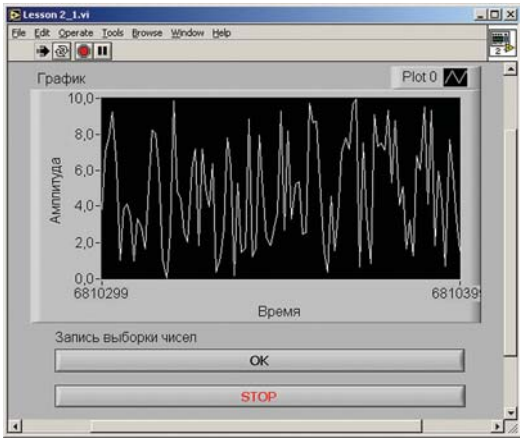


Переходим на интерфейсную панель и запускаем программу на выполнение в циклическом режиме.

Задание: *проверить правильность выполнения поставленной задачи. При выключенном тумблере выставляемое значение должно умножаться на 10, а при включенном не должно изменяться*

Попробуем теперь создать программу, где будут использоваться приемы и навыки программирования, рассмотренные как в этом, так и в предыдущих уроках. Задача состоит в том, чтобы написать программу, которая генерирует последовательность случайных чисел и выводит их в виде графика. Кроме того, по однократному нажатию на кнопку должна производиться запись выборки чисел. Графический интерфейс должен включать, как необходимый и достаточный минимум, три графических объекта: окно отображения графика, кнопку записи данных на диск и кнопку останова выполнения программы.

В результате, должна визуально получиться похожая "лицевая" часть программы:



В процессе выполнения задания воспользуемся дополнительными функциями, связанными с нажатием на интерфейсные кнопки/переключатели. Речь идет о реакции (поведении) кнопки на нажатие, т.е. возвращается ли кнопка после нажатия в исходное состояние или же остается в нажатом состоянии и т.д. В LabVIEW реализовано шесть вариантов реакций на нажатие кнопки:

Switch When Pressed (Реагирует на нажатие);

Switch When Released (Реагирует на отжатие);

Switch Until Released (Реагирует на нажатие и отжатие);

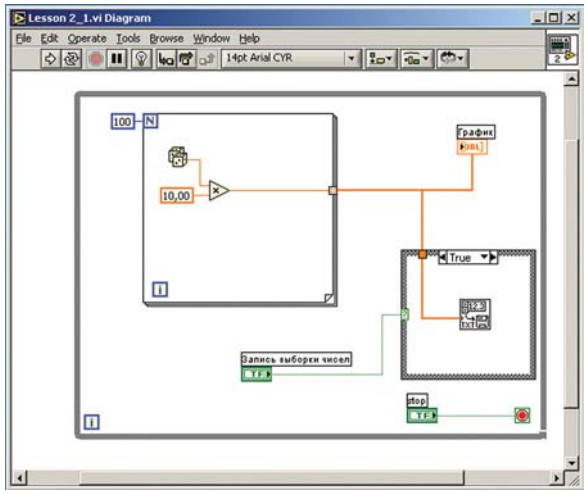
Latch When Pressed (Изменяет управляемое значение при нажатии и обеспечивает автовозврат не раньше, чем произойдет чтение в программе);

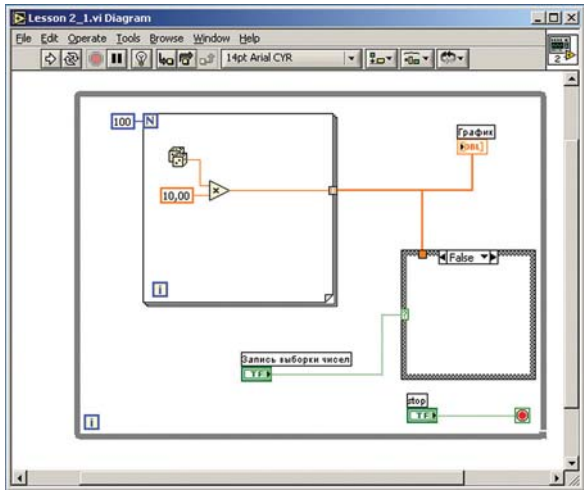
Latch When Released (Изменяет управляемое значение только после отпускания кнопки и обеспечивает автовозврат);

Latch Until Released (Изменяет управляемое значение при нажатии и обеспечивает автовозврат не раньше, чем произойдет чтение в программе или будет отпущена кнопка). В нашей программе нужно установить для обеих кнопок такие опции реакции на нажатие, чтобы кнопки возвращались в предыдущее положение. Если этого не сделать для первой кнопки (записи данных на диск), то после нажатия на кнопку во время работы программы, будет циклически "выскакивать" меню записи файла на диск, т.е. в результате получится, что осуществить остановку программы будет невозможно.

Для реализации этого небольшого нововведения необходимо подвести указатель мыши к одной из кнопок и нажать правую кнопку мыши. В выпадающем меню выбрать ***Mechanical Action***, а в нем ***Latch When Released***. Такую же операцию можно проделать и для кнопки останова. Для освоения различных режимов работы, самостоятельно поэкспериментируйте при запуске и выполнении программы.

Теперь остается реализовать алгоритм программы. В результате получим следующую диаграмму:





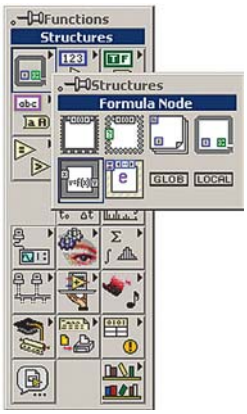
Из приведенных состояний программы видно, что генерируются выборки случайных чисел (по 100 значений в каждой), результат выводится в виде графика, а запись осуществляется посредством нажатия на кнопку. Следует отметить, что реализация ***CASE***-структуры выполнена так, что при нажатии (т.е. когда условие "**True**"), вызывается диалоговое меню записи файла на диск и осуществляется запись данных. Если же условие "**False**", то запись не происходит.

В некоторых ситуациях обычная синтаксическая запись выражения гораздо более удобна и выглядит нагляднее, чем эквивалентная графическая диаграмма. Особенно это актуально при написании математических выражений, вычисляемых по определенной формуле. Для этой цели в LabVIEW существует механизм ***Formula Node*** (Формульный узел или Редактор Формул). ***Formula Node*** относится к еще одной разновидности структур. Поэтому он находится в ***(Functions)*** >> ***Structures*** >> ***Formula Node***.

Примечание: *Очень интересная и полезная реализации структуры выбора. Следует взять на заметку!*

Напишем новую программу, которая будет строить графическую зависимость амплитуды от текущего значения переменной цикла. Редактор формул должен будет реализовывать следующую зависимость:

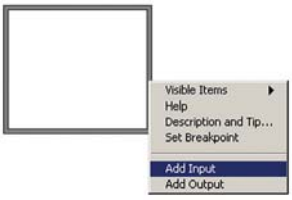
Результат = Амплитуда * SIN (Константа * Текущее значение переменной цикла)



Установим ***Formula Node*** в окно редактирования диаграмм. Для того чтобы можно было использовать этот "черный квадрат", нужно подвести указатель мыши к левой стороне прямоугольника и нажать правую кнопку мыши.

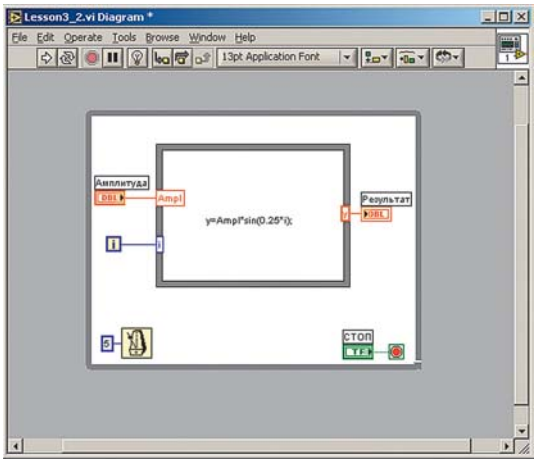
В выпадающем меню выбрать опцию ***Add Input***, которая добавляет входной параметр узла. Ввести имя входного параметра ***"Амплитуда"*** (только латинские символы), например, ***Ampl.***

Аналогично добавляем еще один входной параметр, соответствующий переменной цикла. Называем его ***i***. Для добавления выходного параметра необходимо нажать правой клавишей мыши на правой стороне рамки редактора формул, выбрать опцию ***Add Output*** и ввести имя выходного параметра ***y***.



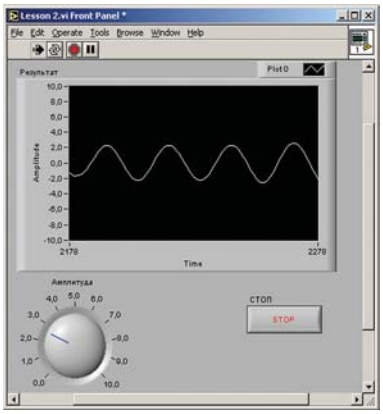
Чтобы ввести формулу, необходимо сначала переключиться в режим редактирования текста: ***Tools*** >> ***Edit Text***, затем установить курсор внутрь окна редактора формул и ввести ее. Для размещения формулы в середине окна редактора формул нужно перед первым символом нажать несколько раз ***Enter***.

Создаем диаграмму, т.е. устанавливаем необходимые компоненты и делаем соответствующие связи.



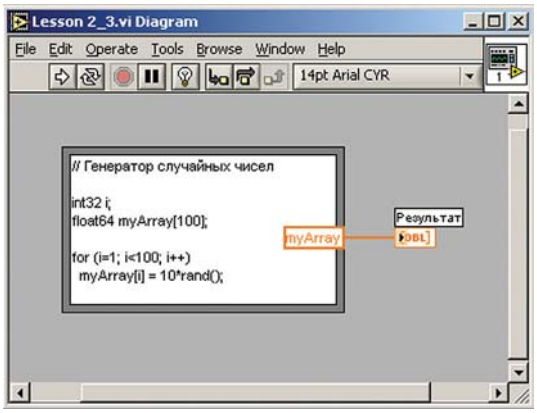
Графический интерфейс составьте по своему усмотрению. Запускаем программу на исполнение.

Вывод: *использование редактора формул в программах дает возможность реализации отдельных функциональных блоков в более компактном виде.*



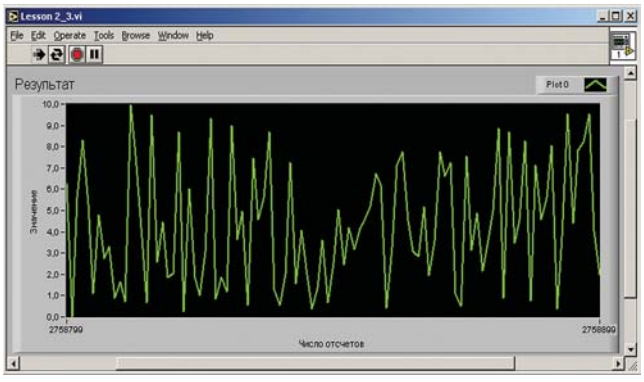
Усвоив принципы использования редактора формул, составим программу генерации последовательности случайных чисел. Синтаксические конструкции, которые используется в таких блоках, позаимствованы из языка Си, что упрощает их понимание и использование.

Из приведенной ниже графической диаграммы видно, что синтаксис записи редактора формул очень похож на код языка Си. Во фрагменте программы использованы комментарии, объявления переменной и массива, а также тело самой программы.



Следует также обратить внимание на то, что параметрами редактора формул могут быть не только переменные или константы, но и другие типы данных, например, как в нашем случае - массив.

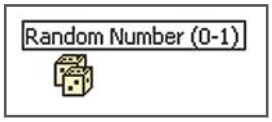
Написав такую программу и запустив ее на выполнение, получим сгенерированную последовательность из 100 случайных чисел:



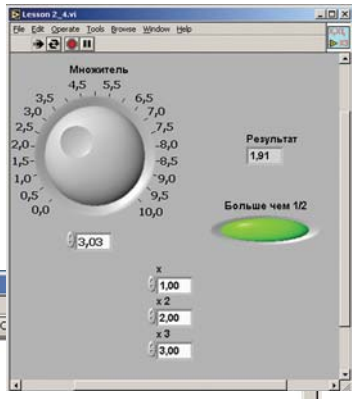
Создавая свои программы, в которых реализуются самые различные алгоритмы, со временем Вы непременно столкнетесь с тем, что диаграмма занимает очень много места, выходя за рамки окна редактирования, и как следствие этого, теряется читаемость. Также очень часто возникает такая ситуация, что используется один и тот же алгоритм (фрагмент программы) несколько раз, тем самым, увеличивая размеры программы.

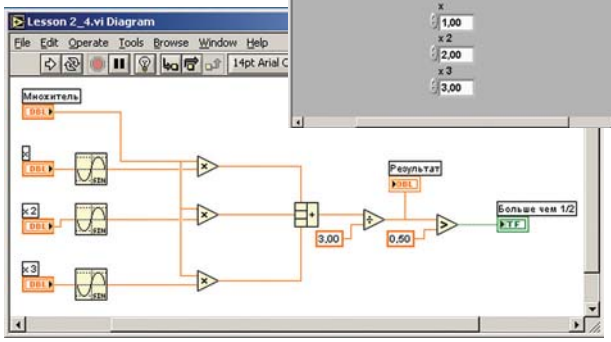
Как и в любом другом языке программирования, в LabVIEW реализована возможность использования подпрограмм (SubVI) для часто повторяющихся фрагментов кода или для функционально независимых от главной программы блоков.

В предыдущем уроке мы уже использовали подпрограммы, например, когда мы устанавливали иконку генератора случайных чисел. Другими словами, эту иконку можно рассматривать и как подпрограмму, так как "внутри" нее реализован алгоритм генерации случайного числа. Т.е. это подпрограмма, в которой отсутствуют какие-либо входные параметры, однако есть один выходной - сгенерированное случайное число.

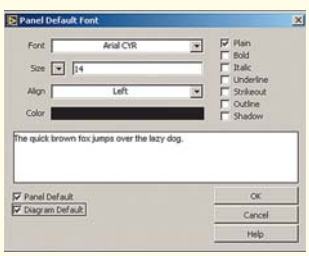


Из всего вышесказанного можно сделать вывод, что подпрограмма LabVIEW - это иконизированное представление какого-либо алгоритма со своим графическим интерфейсом или без него. Создание подпрограммы по своей сути не будет отличаться от написания тех программ, которые мы составляли раньше. Составим подпрограмму (графическую диаграмму и интерфейс) как показано на приведенных ниже рисунках:





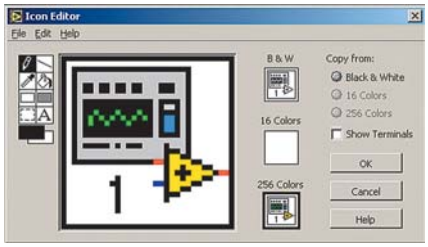
Совет: Если Вы используете LabVIEW в среде Windows2000/XP, то, возможно, установленный по умолчанию шрифт не поддерживает вывод русскоязычных символов. Это проявится, когда вместо русских букв в подписях используемых элементов, Вы увидите "абракадабру". Не смущайтесь, выделив надпись и выбрав в выпадающем меню редактирования текста (расположено на линейке инструментов) кирилизированный шрифт (например, **Arial CYR**), Вы решите проблему. Для перевода сразу всех надписей, предварительно требуется выделить всю область передней панели или диаграммы. Кроме того, используя меню диалога выбора шрифта по умолчанию (**Font Dialog**), можно навсегда забыть о такого рода трудностях.



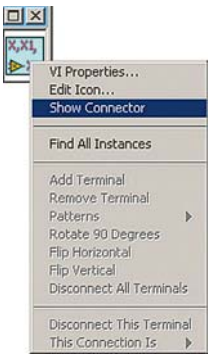
Входными параметрами в нашем примере являются четыре действительных числа, три из которых - основные параметры, а четвертый - множитель. Каждое входное значение умножается на множитель, полученные значения усредняются, т.е. суммируются и делятся на три. Дальше осуществляется элементарная проверка. Если получаемое значение больше чем 0.5, тогда "зажигается" лампочка индикатора (значение "true").

Теперь создадим уникальную графическую иконку для нашей подпрограммы. Для реализации этого необходимо подвести указатель мыши к установленной по умолчанию иконке в правом верхнем углу интерфейсной панели и сделать двойной щелчок левой клавишей мыши. Появится графический редактор иконок со стандартным набором инструментов для создания желаемого образа.

Надеемся, что у Вас получится более "интересное" изображение. Нажав OK, увидим, что иконка в правом верхнем углу изменится на желаемую.



Теперь остается сделать еще один очень важный шаг для создания и использования подпрограммы, а именно - установить соответствия между функциональными элементами программы и выводами иконки. Для этого необходимо подвести указатель мыши к изображению иконки в правом верхнем углу интерфейсной панели, и нажать правую клавишу мыши. В выпадающем меню следует выбрать опцию ***Show Connector***. После этого действия иконка изменится и появится несколько, в нашем случае шесть, клеток (клемм), которые и будут соответствовать нашим шести параметрам подпрограммы (4-входных и 2-выходных). Чтобы установить соответствие параметров подпрограммы выводам коннекторов, необходимо произвести следующие операции. Подвести указатель мыши в виде соединительной катушки к первому контакту на иконке коннектора и нажать левую клавишу мыши. Активная клеточка подсветится. Далее следует подвести указатель к первому параметру подпрограммы, а именно к "x", и нажать левую клавишу мыши. Цвет клеточки изменится. Для остальных параметров следует проделать аналогичную операцию, последовательно устанавливая соответствия между графическими элементами и клеммами иконки коннектора.



Следует отметить, что цвет клеммы зависит от типа данных элемента. Так, если элемент принимает значения действительного типа, то клемма будет оранжевая, если же это булевый тип - то зеленая. Т.е., каким цветом отображаются эти элементы в окне редактирования диаграмм, такого же цвета и клеммы.

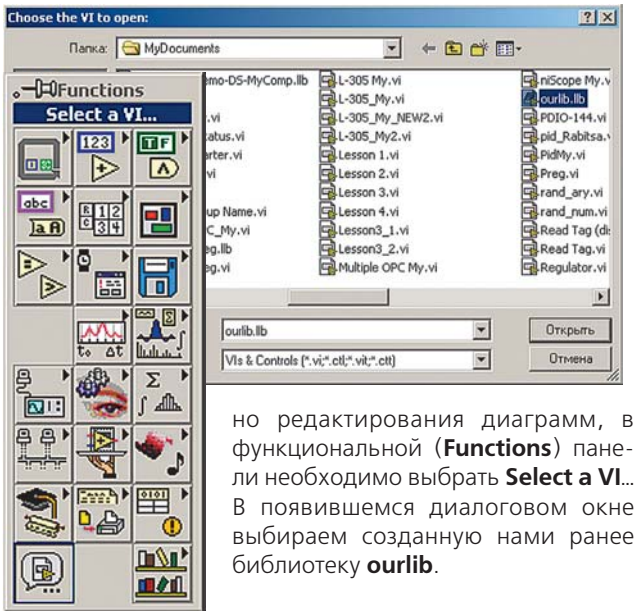
Установив все необходимые соответствия, можно сохранить подпрограмму. Особенностью подпрограмм LabVIEW является то, что они могут функционировать как независимые автономные приложения, так и быть частью основной программы.

Следующим шагом будет сохранение нашей подпрограммы в библиотеке. Библиотека LabVIEW представляет собой составной файл, в котором находятся несколько различных виртуальных инструментов (как правило, объединенных тематически).

Выбрать в меню ***File*** опцию ***Save As...*** В появившемся меню нужно выбрать желаемую директорию и нажать кнопку ***New VI Library***. Далее следует ввести имя библиотеки, например "***ourlib***". Потом следует ввести имя самой подпрограммы, например "***subsin***".

Теперь перейдем к созданию главной программы, в которой будет использоваться написанная нами подпрограмма. Есть несколько путей вызова иконки подпрограммы в окно редактирования диаграмм:

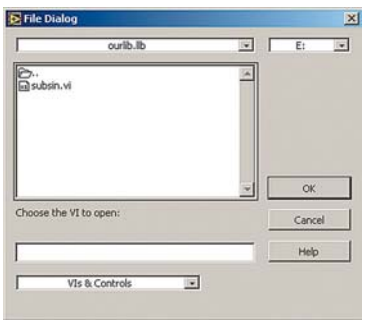
- Первый заключается в простом перетягивании иконки, которая находится в правом верхнем углу открытого окна подпрограммы в окно редактирования диаграмм главной программы. Нажав левую клавишу мыши на этой иконке, и удерживая ее в нажатом состоянии, перетащите в окно редактирования диаграмм.
- Второй способ более универсальный. Перейдя в ок-



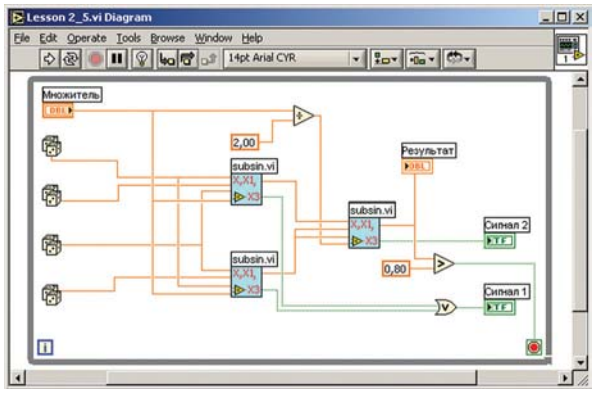
но редактирования диаграмм, в функциональной (**Functions**) панели необходимо выбрать **Select a VI**... В появившемся диалоговом окне выбираем созданную нами ранее библиотеку **ourlib**.

Примечание. *Возможно сохранение подпрограммы сразу как файла с расширением ***.vi.** Однако файлы библиотек **(*.llb)** позволяют упорядочить различные виртуальные инструменты по категориям и избежать нагромождения и путаницы при выборе **VI**.*

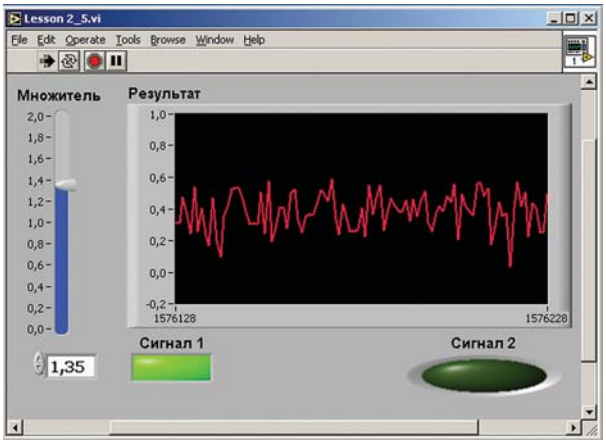
В появившемся окне выбрать **subsin** и установить иконку в произвольное место окна редактирования диаграмм.



Составьте графическую диаграмму, как показано на рисунке ниже.



Суть программы состоит в том, что на входы подпрограмм подаются случайные числа от четырех источников. Далее они обрабатываются и анализируются в соответствии с диаграммой. Результат выводится в виде графической зависимости. Условием завершения программы является превышение выходным параметром значения 0,8.



Задание. *Самостоятельно разобраться, как работает программа, изменяя как значения параметров, так и алгоритм.*

Пусть данный пример и не решает какую либо прикладную задачу, однако он позволяет понять необходимость использования подпрограмм и помогает усвоить азы применения ***Sub VI***.

По аналогии, для главной программы также можно создать свою иконку, установить соединения и использовать ее уже как подпрограмму в других приложениях.

Не сомневаемся, что созданные Вами на этом уроке программы успешно заработали. Это означает, что Вы уже достигли определенного профессионального уровня программирования в LabVIEW.

В качестве домашнего задания рекомендуем Вам освоить технологию оформления документов LabVIEW: не только интерфейсная панель, но и "исходный текст" программы,т.е. диаграмма, должны выглядеть красиво.

Авторы – сотрудники "ХОЛИТ Дэйта Системс", г. Киев

КОНТАКТЫ:
т. (044) 241-8739, 241-6754
e-mail: info@holit.com.ua