Уроки по LabVIEW

На этом уроке Вы научитесь:

- *использовать в своих программах структуры типа "последовательность"*
- *создавать различными способами массивы*
- *использовать основные функции для работы с массивами*
- *применять свойства полиморфизма при работе с массивами*
- *использовать новый тип данных – кластеры*
- *строить несколько зависимостей на одном графике*

При создании VI последовательность выполнения двух независимых фрагментов программы не определена, что в некоторых случаях приводит к неоднозначному результату. Для решения этой проблемы существует специальная структура – **Sequence** *(Последовательность)*. По своему принципу действия она напоминает киноленту, когда последовательно выводятся на экран отдельные кадры. Таким образом, **Sequence** определяет порядок выполнения фрагментов программы.

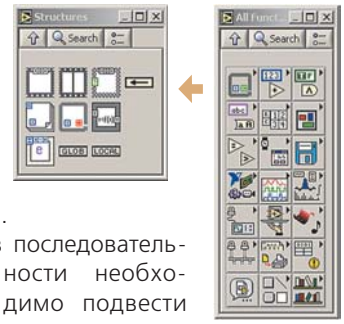
Покажем принцип работы такой структуры. Для этого напишем программу, которая будет подсчитывать время выполнения определенного цикла. В программе будем использовать последовательность с тремя кадрами.

Будут обеспечены следующие функции:

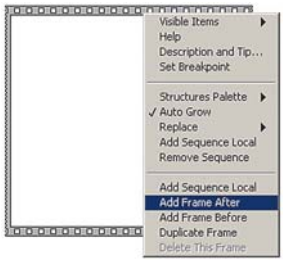
- первичная инициализация системного времени;
- фрагмент программы, время выполнения которого и подсчитывается;
- получение значения системного времени;
- подсчет времени работы программы;
- перевод значения из миллисекунд в секунды.

Будем реализовывать программу шаг за шагом. Первый шаг – создание нового приложения **File≫New VI**. Далее следует переключиться в окно редактирования диаграмм.

Следующим шагом будет выбор последовательности-структуры из функциональной палитры: **Functions≫Structures≫Sequence** и перетягивание ее в область редактирования диаграмм.



Для создания кадров последовательности необходимо подвести указатель мыши на границу области структуры и нажать правую кнопку мыши. В появившемся меню необходимо выбрать **Add Frame After** *(Добавить кадр после)*. Таким образом, создадим 3 кадра (0...2).

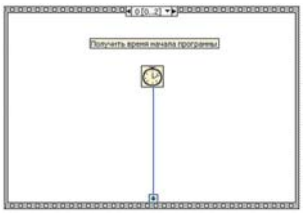


Переменные, которые используются для передачи данных между кадрами – **Локальные Переменные Последовательности** *(Sequence locals)*. В нашей программе такая переменная будет использоваться для передачи значения системного времени.

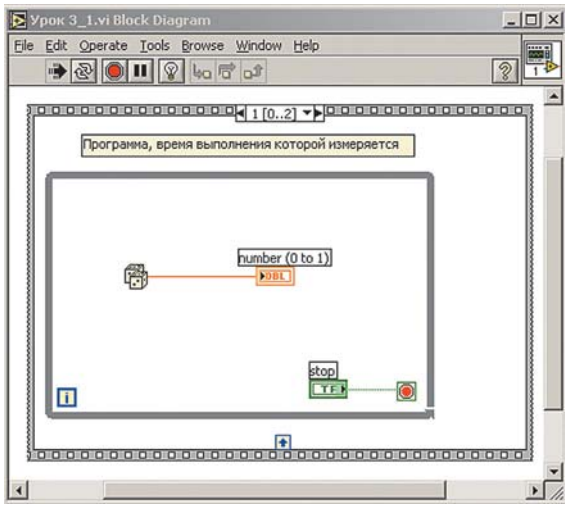
Создадим "наполнение" для каждого кадра. В начальный (0-й) кадр поместим компонент **Tick Count**, который считывает текущее значение системного таймера и возвращает результат в миллисекундах. Эта подпрограмма может быть загружена из меню **Functions≫Time & Dialog≫Tick Count (ms)**.

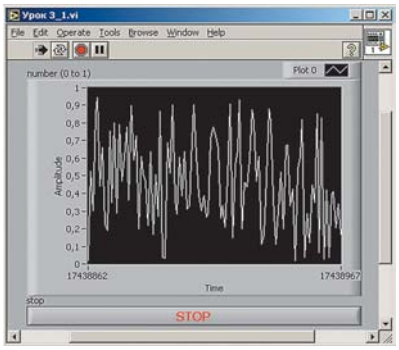
Далее создадим упомянутую выше **Локальную Переменную**. Для этого нужно подвести указатель мыши к границе структуры, нажать правую кнопку мыши, и в выпадающем меню выбрать **Add Sequence Local** *(Добавить Локальную Переменную Последовательности)*.

Соединяем вывод **Tick Count (ms)** с появившимся терминалом локальной переменной. В результате, внутри него появится стрелочка, указывающая на то, что данные поступают из текущего кадра.



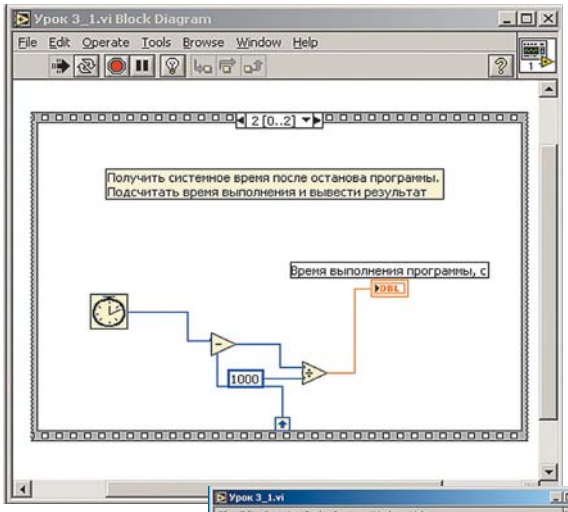
"Наполним" содержанием следующий (1-й) кадр. В нем реализуем фрагмент программы, для которого подсчитывается время работы. Это будет, как и в предыдущих примерах, генератор случайных чисел.

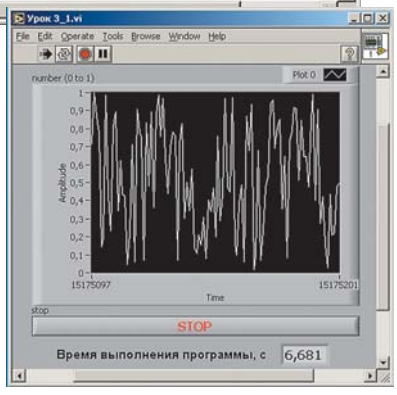




Программа реализуется в виде цикла **While-Loop**, условием выхода из которого является нажатие кнопки останова. График генерации случайных чисел выводится на переднюю панель в виде графика **Waveform Chart**.

В последнем (2-м) кадре воспользуемся все тем же Tick Count (ms) и подсчитаем разницу во времени. Для этого сравним текущее значение времени со значением, полученным в 0-м кадре. Использовать значение, полученное в первом кадре, можно соединив Локальную Переменную (стрелочку в квадратике) с соответствующим выводом. Переводим миллисекунды в секунды путем деления значения на 1000 и выводим результат на цифровой индикатор, предварительно установив его на интерфейсной панели.

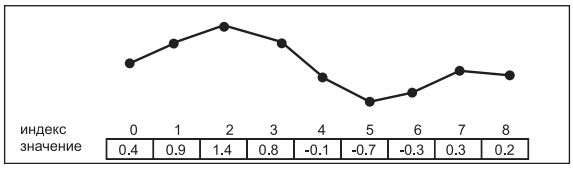




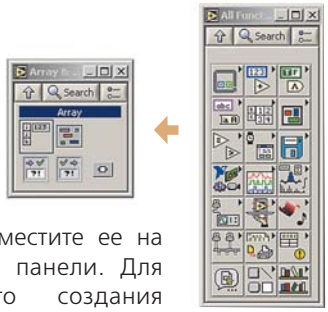
Остается проверить программу на работоспособность. Запускаем ее на выполнение. Для останова нажимаем кнопку "STOP". В результате мы увидим значение времени в секундах, которое было затрачено на выполнение программы.

Перейдем к рассмотрению массивов, как важного и часто используемого элемента программирования задач сбора данных и управления. Массив - набор данных одного типа. Массив может быть как одномерным, так и многомерным, и иметь до 231 элементов (ограничивается объемом оперативной памяти).

Массивы в LabVIEW могут быть любого типа. Доступ к произвольному элементу осуществляется через его индекс. Индекс принадлежит диапазону чисел 0..N-1, где N – количество элементов массива. Заметьте, что первый элемент имеет индекс 0, второй - 1, и т.д. Структуру одномерного (1D) массива можно представить как показано рисунке:



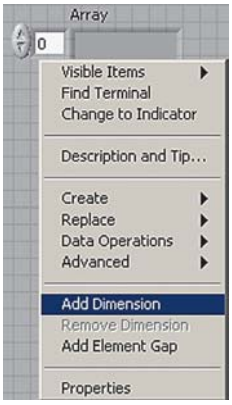
Создание массива, как элемента управления или индикации, осуществляется комбинированием оболочки массива и объекта данных, который может быть цифровым, булевым, строковым или комбинированным (кластером). Давайте создадим массив.

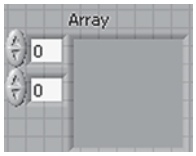


Первый шаг: Выберите пустую оболочку массива **(Array)** из палитры **Controls≫ Array&Cluster≫Array** и поместите ее на интерфейсной панели. Для окончательного создания массива перетяните объект данных определенного типа вовнутрь оболочки массива: **Controls≫Numeric≫Digital Control**.

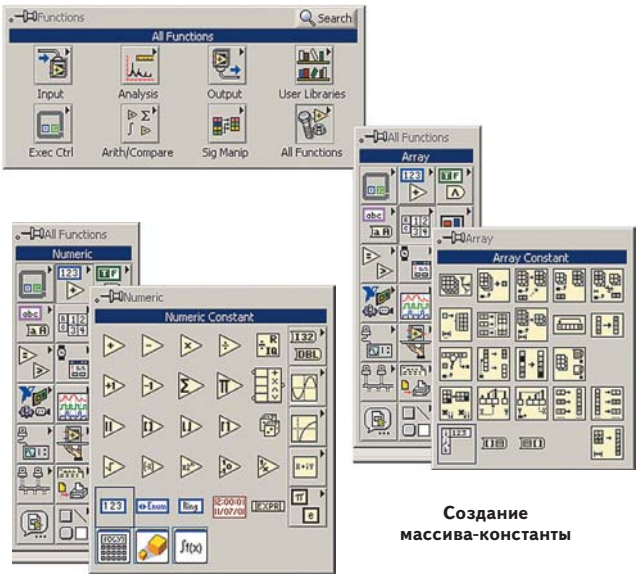
В отличие от одномерного массива, двумерный (2D) массив имеет два индекса для каждого элемента. Первый указывает на номер строки, а второй - номер столбца.

Для создания 2-мерного массива данных из 1-мерного необходимо подвести к нему указатель мыши и нажать правую кнопку. Затем, в выпадающем меню, выбрать **Add Dimension** (*Добавить размерность*). В результате внешний вид элемента управления двумерного (2D) массива на передней панели Вашего Виртуального Инструмента примет вид:



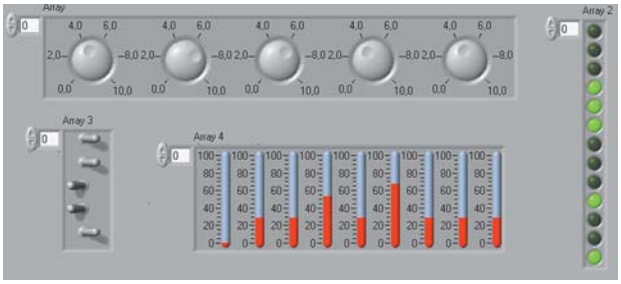


Вы также можете создать массив-константу, например, для задания набора коэффициентов. Для этой цели необходимо выбрать **Array Constant** в палитре **Functions≫ Array** и поместить в окно редактирования диаграмм. Далее, в созданную оболочку массива необходимо поместить константу желаемого типа данных, например, **Numeric Constant** из **Functions≫Numeric**.

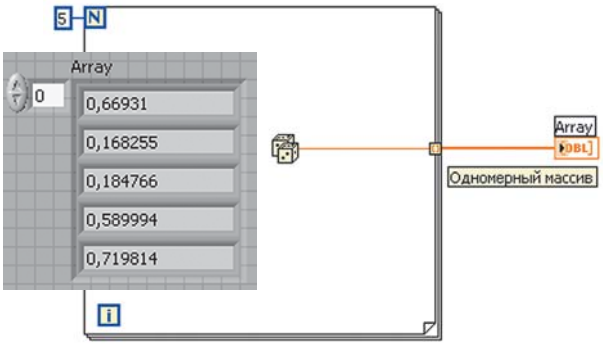


Создание массива-константы

Рассмотрим пример массивов элементов управления и индикаторов, различных типов данных:



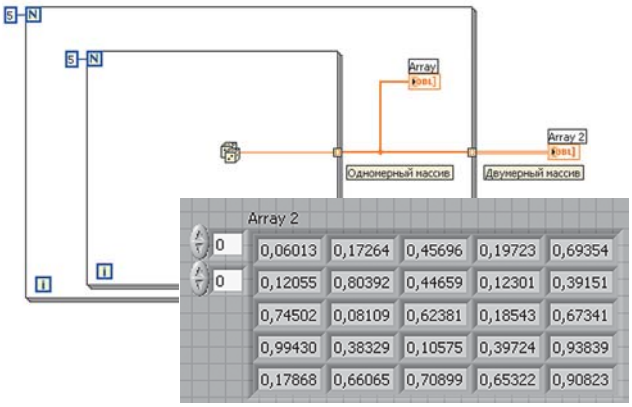
В предыдущих уроках мы уже использовали структуры типа "цикл" для создания массивов (вспомните пример с записью на диск последовательности случайных чисел). Циклы типа **For Loop** и **While Loop** могут индексировать и создавать массивы автоматически. Такая особенность называется - автоиндексирование. Приведенная ниже диаграмма показывает как происходит создание и индексирование массива при использовании цикла типа **For Loop**.



Созданный массив выводится на индикатор. Заметим также, что соединительная линия внутри цикла меняется на утолщенную за его пределами, что сигнализирует о том, что мы уже имеем массив.

Для создания 2-мерного массива необходимо созданную структуру поместить еще в один внешний цикл.

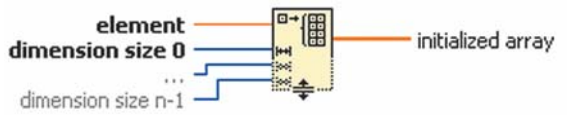
В результате на выходе первого (внутреннего) цикла будет сформирован одномерный массив из пяти элементов. А на выходе внешнего цикла - образуется двумерный массив 5x5, где 5-количество рядов, а 5-количество столбцов результирующего массива.



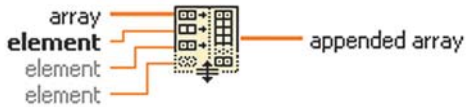
В LabVIEW реализовано достаточно много различных функций для работы с массивами в окне редактирования диаграмм. Они находятся на функциональной панели (Functions) в разделе массивов (**Array**). Мы рассмотрим только некоторые из них - наиболее важные функции.

Array Size – возвращает размер массива. Если входной массив N-мерный, то выходной параметр (**size**) это одномерный массив с N элементами, которые указывают его размерность. Например, если входной 2D массив имеет размер 4x6, то на выходе будет сформирован одномерный массив из двух чисел, соответственно 4 и 6.

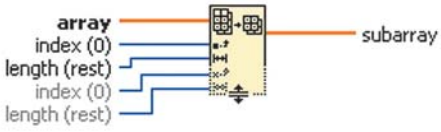
Initialize Array – создает массив размерности **dimension size**, все элементы которого принимают значение **element**. Для создания многомерного массива необходимо "растянуть" иконку - "потянуть" за правый нижний угол вниз.



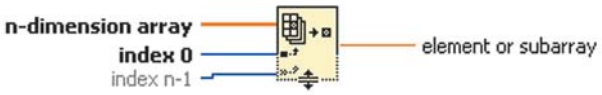
Build Array – создает массив из массивов или элементов, или добавляет элементы к уже существующему массиву. Вы можете изменить размер пиктограммы этой функции для увеличения количества входов.



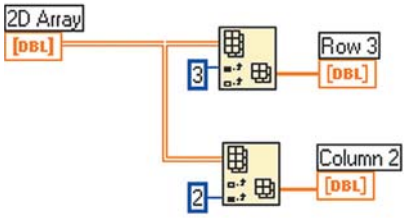
Array Subset – "вырезает" подмассив согласно заданным значениям стартового индекса (**index**) и длине (**length**).



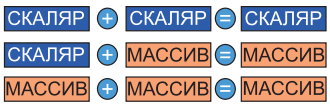
Index Array – выделение элемента массива (доступ к элементу массива). Используя эту функцию можно выделять не только элементы массива, но и желаемые ряды и столбцы массивов.



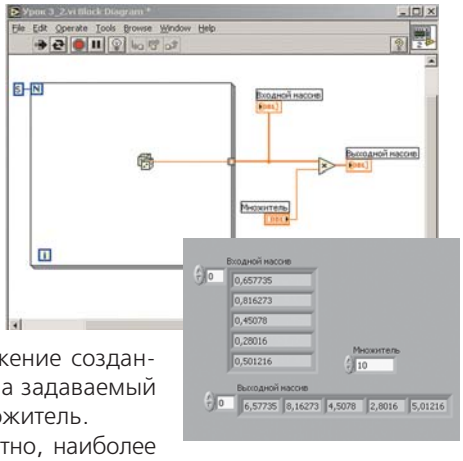
На приведенной диаграмме показано, как из существующего двумерного массива выделяется 3-я строка (в первом случае) и 2-ой столбец (во втором случае).



В LabVIEW для работы с массивами реализованы такие арифметические функции как сложение, умножение, деление и другие. Эти функции являются полиморфными, поэтому входные данные могут быть разного типа - скалярными или массивами. Например, можно суммировать скаляр с массивом или сложить два массива вместе.



На приведенном примере показано, как легко можно умножить каждый элемент массива на заданное число. Для этого генерируется массив, элементы которого-случайные числа. Далее реализуется умножение созданного массива на задаваемый параметр - множитель.

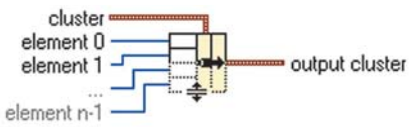


Как известно, наиболее удобной формой представления массива данных является их представление в виде графиков. В предыдущих уроках мы уже встречались с простейшим выводом массива данных на графический индикатор. В этом уроке мы рассмотрим использование графических индикаторов совместно с массивами. Как правило, нам необходимо изображать зависимость одного параметра от другого, т.е. 2D-графики. Это могут быть зависимости такого рода как, например, амплитуда сигнала от времени, спектр сигнала, статистические распределения и т.д.

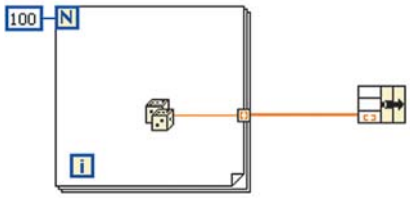
Однако часто встречаются ситуации, когда необходимо задать дополнительные параметры, такие как начальное смещение "0" и интервал. Для решения этой проблемы в LabVIEW существует структура данных, называемая **Кластером (Cluster)**.

Кластер – это связанная структура данных, элементы которой могут быть разного типа (аналог **struct** в "C"). Т.е., Вы можете сформировать структуру, элементами которой будут, например, целое число, действительное число, строка и др.

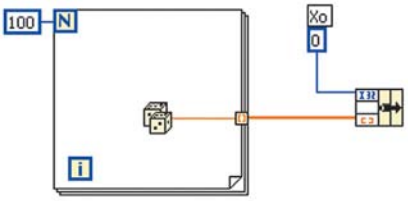
Для связки данных используется функция **Bundle (Связать)**, которая находится в **Functions ≫ Cluster ≫ Bundle**. В нашем случае будут компоноваться в кластер массив данных (т.е. данные, которые откладываются по оси Y), начальное значение по оси X, и шаг по оси dX.

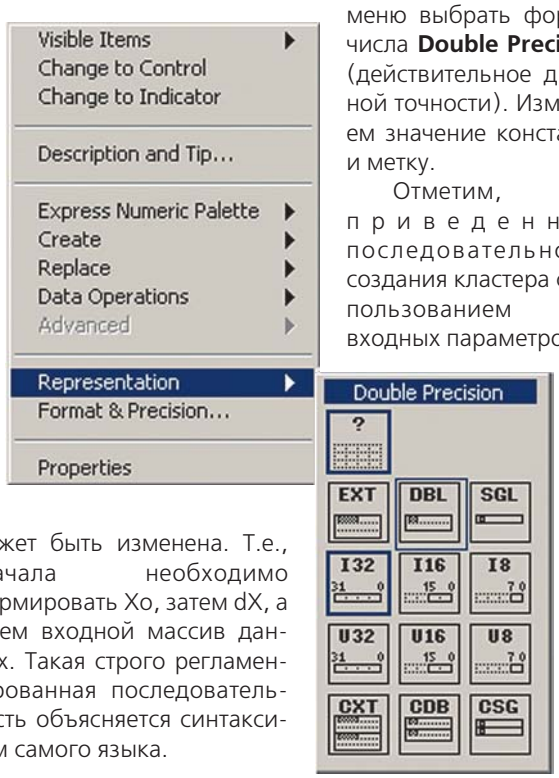


Рассмотрим построение графической зависимости, используя функцию связки **Bundle**. Как получить массив случайных чисел- мы уже знаем. Следующим шагом будет формирование кластера. Для этого нужно установить в окне редактирования диаграмм функцию связки **Functions ≫ Cluster ≫ Bundle**. Добавим еще один вход **Связки**. Подводим стрелку мыши к пиктограмме, нажимаем левую клавишу и, не отпуская ее, потянем вниз так, чтобы получилось три входа. Свяжем пиктограмму с данными от генератора случайных чисел. Далее создадим две константы **(Functions ≫ Numeric ≫ Numeric Constant)**, которые задают начало отсчета по оси X и шаг. Для этого установим первую константу. Изменим ее метку на Xo, значение с 0 на 3, и соединим с верхним входом связки.



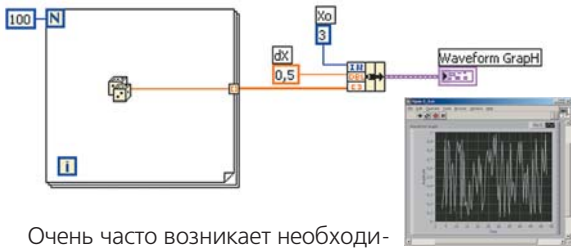
Создадим еще одну константу, которая будет задавать шаг по оси X. Назначим имя константы dX, а значение как 0,5 и соединим со свободным вводом функции связки. Если значение интервала вывода точек не является целым числом, то нам необходимо сначала изменить формат константы шага. Для этого нужно подвести указатель мыши к вновь созданной константе, нажать правую клавишу мыши, затем выбрать пункт меню **Representation**, а в подменю выбрать формат числа **Double Precision** (действительное двойной точности). Изменяем значение константы и метку.





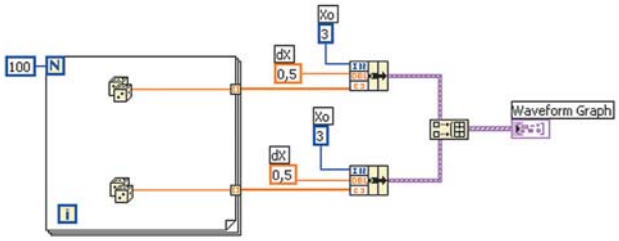
Отметим, что приведенная последовательность создания кластера с использованием трех входных параметров не может быть изменена. Т.е., сначала необходимо формировать Xo, затем dX, а затем входной массив данных. Такая строго регламентированная последовательность объясняется синтаксисом самого языка.

И последний шаг - установим **Waveform Graph** на интерфейсную панель и соединим с выходом функции **Bundle**. В результате должна получиться диаграмма:

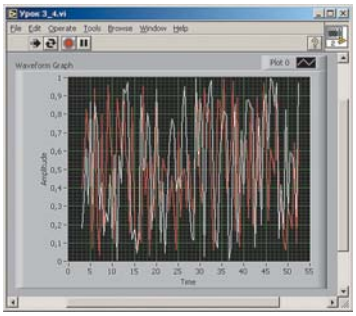


Очень часто возникает необходимость построения одновременно нескольких зависимостей на одном графике - для большей наглядности, при визуальном сравнении выводимых данных и т. д. Для того, чтобы реализовать возможность такого построения, необходимо воспользоваться уже известной нам функцией **Build Array (Functions ≫ Array ≫ Build Array)**.

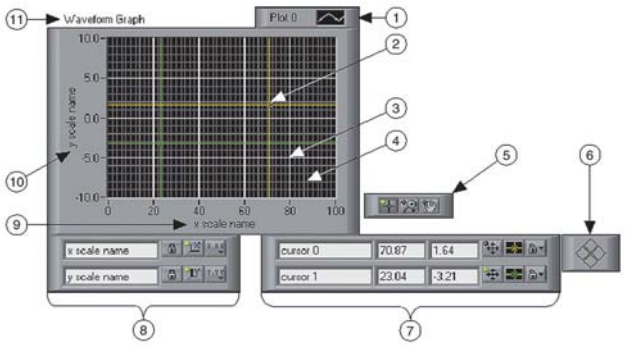
Модифицируем предыдущую программу - генерируем второй массив случайных чисел. "Перетягиваем" в окно редактирования диаграмм функцию **Build Array**. Меняем ее размер так, чтобы образовалось два входа и делаем соединения, как показано на диаграмме:



После запуска программы убеждаемся, что действительно на одном графике размещены две различные зависимости.

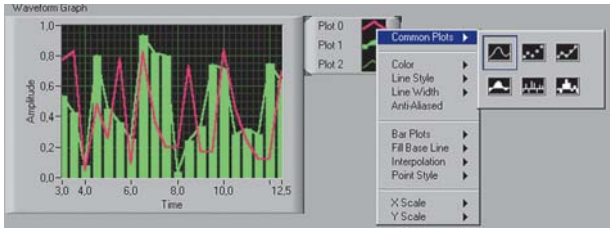


При нажатии правой клавиши мыши на области построения зависимостей появляется выпадающее меню с множеством различных опций и настроек. Причем, производя нажатия на различных компонентах, появляются уникальные, свойственные именно этим элементам, опции настроек. Так если сделать визуально видимыми все основные компоненты **Waveform Graph**, то это будет выглядеть следующим образом:



Как видим, имеется достаточно много инструментария, который позволяет существенно изменять выводимые графики.

① **Plot Legend** *(настройка свойств графика)* – определяет цвет и стиль графика или графиков. Изменение этой группы параметров позволяет индивидуально настраивать свойства каждой выводимой зависимости.



② **Cursor** *(курсор)* – показывает положение курсора для каждой выводимой зависимости отдельно.
③ **Grid mark** – крупная сетка.
④ **Mini-grid mark** – мелкая сетка.
⑤ **Graph palette** *(палитра графика)* – позволяет манипулировать построением зависимости, например масштабом, во время работы программы.
⑥ **Cursor mover** – перемещает курсор в задаваемую позицию.
⑦ **Cursor legend** *(тип курсора)* – управляет и модифицирует опции, связанные с курсорами.
⑧ **Scale legend** *(тип шкал)* – осуществляет конфигурацию шкал.
⑨ **X-scale** – шкала X.
⑩ **Y-scale** – шкала Y.
⑪ **Label** – метка *(уникальное имя индикатора)*.

"Пощупать" возможности графического представления данных мы предлагаем Вам самостоятельно в качестве домашнего задания.

Уроки по LabVIEW

На следующем уроке:

- *изменение свойств элементов управления и индикаторов при работе со строками*
- *преобразование чисел в строки*
- *визуализация записи на диск данных в виде таблиц и графиков, работа с "HELP"*
- *создание приложений, которые могут реализовывать самый широкий спектр задач, интегрированных в одной LabVIEW-программе*

СЛУЖБА НОВОСТЕЙ

"Образовательные, научные и инженерные приложения в среде LabVIEW и технологии National Instruments"

Под таким названием в России прошла конференция, в которой приняли участие более 200 специалистов, в т.ч. и с Украины. Участники конференции высоко оценили возможности комплекса LabVIEW для решения задач автоматизации измерений, надежность и производительность аппаратных средств National Instruments, а также отметили успешную деятельность компании по продвижению новых технологий в образование, науку и промышленность. Необходимость обучения специалистов, призванных решать сложные задачи автоматизации эксперимента и производства, актуальность внедрения в образование и научные исследования виртуальных технологий была подчеркнута в докладах министра образования РФ В.М. Филиппова и президента РНЦ "Курчатовский институт", академика РАН Е.П. Велихова.

СЛУЖБА НОВОСТЕЙ