Учебный практикум по LabVIEW

Пожалуй, сегодня не нужно объяснять, что такое LabVIEW. Что этот замечательный программный продукт компании National Instruments нужен и исследователю, и инженеру, и студенту, и даже школьнику, что с его помощью эффективно решаются задачи в науке, на производстве и в сфере образования. Во многом благодаря публикациям на страницах журнала "ПиКАД", число пользователей LabVIEW в Украине существенно увеличилось. А вместе с ними увеличилось и число вопросов, требующих более детального освещения. На некоторые из них, а именно – работа с двухкоординатными XY-графиками и основные принципы преобразования цифрового кода, ответы в текущем выпуске. Но вначале обязательная программа – основные математические операции с таким специфическим типом данных, как комплексные числа.

Комплексные числа в LabVIEW представляются константами и переменными специальных типов: "CXT", "CDB", "CSG". Данные типы расположены в нижнем ряду палитры типов ***Representation*** (напомним, доступ к этой палитре возможен из выпадающего при нажатии правой кнопкой мыши на переменную/константу меню). Вам ничего не напоминают данные аббревиатуры? Об их значениях нетрудно догадаться, обратив внимание на второй ряд иконок в палитре типов "EXT", "DBL", "SGL". Да-да, действительно, если во втором ряду находятся иконки типов для дробных чисел - с расширенной, двойной и одинарной точностью (Extended Precision, Double Precision, Single Precision), то в последнем - то же самое для комплексных чисел. Таким образом, комплексное число расширенной точности (Complex Extended, "CXT") представляет собой совокупность двух десятичных дробей расширенной точности - действительной мнимой частей.

В качестве напоминания отметим, что размеры дробных переменных и констант "EXT", "DBL", "SGL" составляют соответственно 16, 8 и 4 байта. Как видим, данные в LabVIEW можно задавать весьма и весьма высокоточно.

Очевидно, что комплексные числа с целочисленными составляющими (если и далее проводить аналогии - то схожие по структуре с I64, I32, U64 и т.д.) применения здесь не находят.

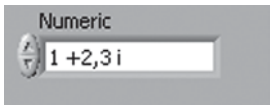
Константы на блок-диаграмме представляются так же, как и в блокнотах инженеров и математиков:

На приведенной диаграмме константа расположена слева. Для изменения любой из ее составляющих просто подведите указатель манипулятора к соответствующему числу и наберите вместо него требуемое.

Следует заметить, что с комплексными числами возможно проведение только некоторых из операций сравнения. Вы можете самостоятельно в этом убедиться, перетаскивая элементы палитры ***Comparison*** и подавая на их входы комплексные аргументы.

В соответствии с понятиями элементарной математики множество комплексных чисел является неупорядоченным, и фразы типа "Комплексное число А больше числа В" смысла не имеют. Поэтому функции "Больше?", "Меньше?", "Больше или равно?" (***Greater?***, ***Less?***, ***Greater Or Equal?***) и прочие здесь "не пройдут". Вы получите сообщение об ошибке по причине несоответствия типов данных.

По этой же причине при вводе комплексного числа с передней панели возможно использование только указателя мыши и цифровых клавиш на клавиатуре. А вот кнопки ***Increment/Decrement***, хоть и находятся по умолчанию слева от цифрового поля, но не работают.



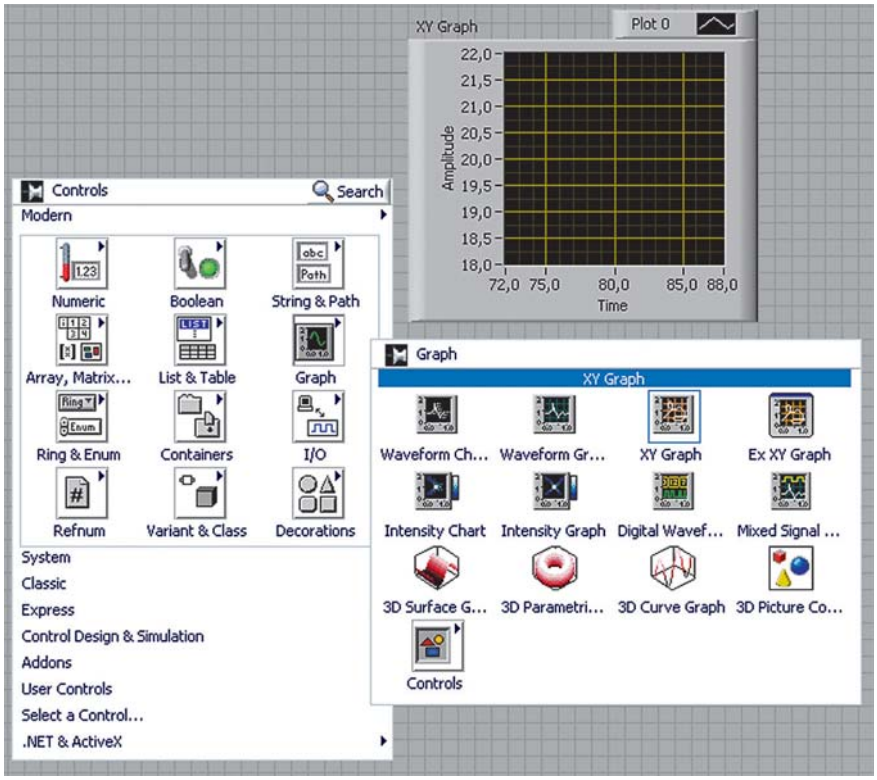
Зато применение функций "Равно?", "Не равно?", "Равно 0?", "Не равно 0?" (***Equal?***, ***Not Equal?***, ***Equal To 0?***, ***Not Equal To 0?***) вполне возможно - ведь комплексные числа являются равными при условии равенства соответствующих им составляющих.

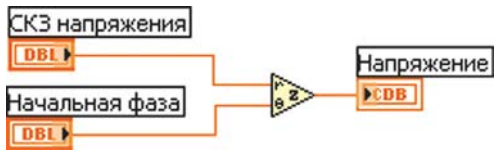
Что касается палитры функций для работы непосредственно с "комплексом", она имеет следующий вид:

Функция ***Complex Conjugate*** уже фигурировала ранее на рисунке с изображением комплексной константы. Принцип ее действия в соответствии с математическими выкладками достаточно прост - смена знака у мнимой части числа. Несмотря на то, что все достаточно просто,

комплексно-сопряженные числа находят широкое применение в математических расчетах, а смена знака вручную занимает достаточно много времени, которое необходимо потратить на "распаковку" числа.

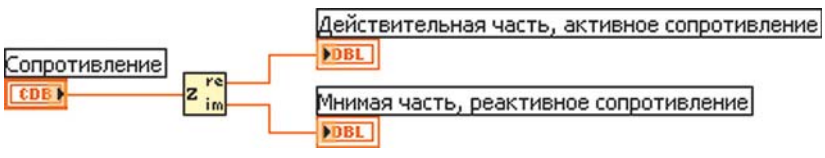
Под распаковкой здесь предполагается выделение из комплексного числа действительной и мнимой частей. А иногда необходимо именно данное разделение. Предположим, нам необходимо рассчитать, какова доля реактивной составляющей сопротивления некоторого проводника либо четырехполюсника. Пускай набор входных данных мы получили путем пропускания через проводник электрического тока с последующим его измерением и падающего на него напряжения. Итак, в условии у нас есть: среднеквадратические значения напряжения и тока, разность фаз между ними. Естественно, для расчета удобно будет задать ток и напряжение комплексными величинами. Для реализации этого выбираем функцию ***Polar To Complex***, формирующую комплексное число при известных модуле и аргументе:



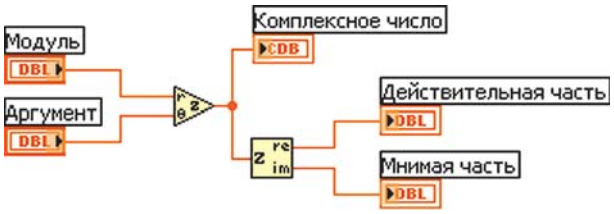


Значение аргумента задается в радианах. Как известно, разность фаз - это разница между мгновенными значениями фаз напряжения и тока. Поэтому записываем эту разность в переменную "Начальная фаза" для напряжения, а для тока соответствующее значение оставляем равным нулю.

Далее в LabVIEW в соответствии с законом Ома делим напряжение на ток и выделяем из полученного значения сопротивления составляющие с помощью функции ***Complex To Re/Im***:



Вот собственно и все, что следует по минимуму знать по комплексным числам в LabVIEW. Подводя итоги, необходимо заметить, что в LabVIEW возможны преобразования в любом направлении между комплексным множеством, декартовыми координаты, полярными координатами. Можете убедиться в этом, изучив действие остальных функций палитры самостоятельно. Также очевидным является то, что сформировать любую из представленных функций можно с использованием других функций палитры. Например, реализация ***Polar To Re/Im*** имеет следующий вид:



В выпуске ПИКАД №1, 2008 уже были рассмотрены общие принципы работы с осциллограммами в LabVIEW. Однако у типа данных ***Waveform*** есть один недостаток: совокупность составляющих осциллограмму данных обязательно должна быть функцией. Вы должно быть помните из курса элементарной математики, что основное ограничение заключается в том, что каждому значению аргумента из области определения должно соответствовать только одно значение функции.

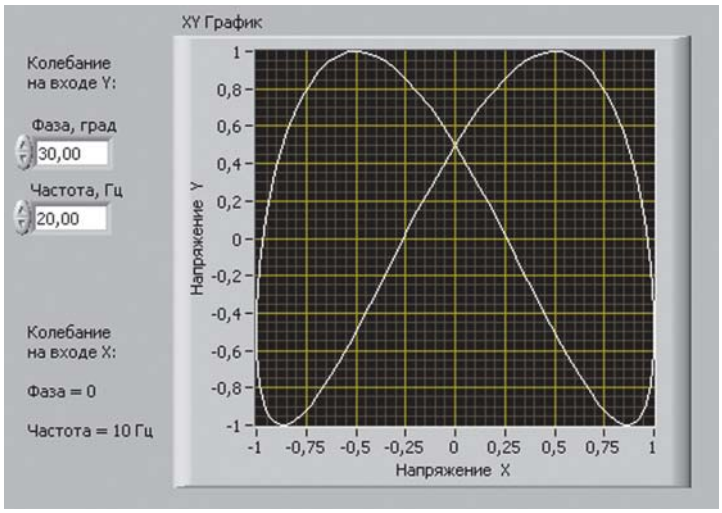
Другими словами, в ***Waveform*** аргументом является время, причем это выборка с постоянным периодом дискретизации, что также не всегда удобно для практических целей. В некоторых случаях для оптимизации полученных от объекта исследования данных этот период меняют, т.е. "выбрасывают" промежуточные неинформативные отсчеты. Можно, естественно, работать с двумя отдельными массивами данных, один из которых - моменты времени, другой, например, полученные значения напряжения. Но, тем не менее, часть перечисленных ограничений можно "обойти". А если требуется построить графическую зависимость между двумя напряжениями, как быть?

Для "безболезненного" решения поставленных задач такого рода в LabVIEW существует специальный тип графического представления данных ***XY Graph***. Одноименный графический индикатор располагается на той же подпалитре, где и ***Waveform Graph***, т.е. поочередно открывая с передней панели LabVIEW палитры ***Controls≫Graphic***.

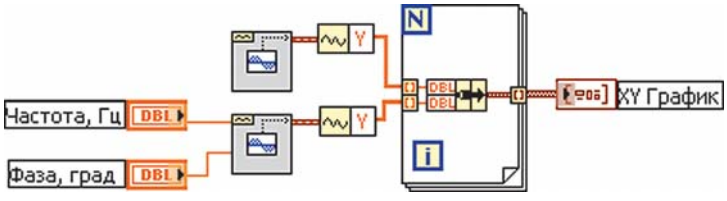
Последовательность построения двухкоординатного графика рассмотрим на следующем примере. Допустим, необходимо снять фазочастотную характеристику четырехполюсника. В этом случае исходными данными будут выборки напряжения на входе и выходе устройства, соответствующие отсчеты которых синхронизированы. Даже простейшие средства сбора данных, такие как многоканальный АЦП платы сбора данных, позволят сформировать необходимые массивы при постановке эксперимента. Если же Вы получили при сборе данных обычные ***Waveform'ы***, то не составит труда "разложить" их на составляющие, используя уже описанную (ПИКАД №1'2008) функцию ***Get Waveform Components***.

Следует помнить, что на вход пиктограммы ***XY Graph*** необходимо подавать массив из кластеров, содержащих по два совпадающих во времени отсчета (по X и Y).

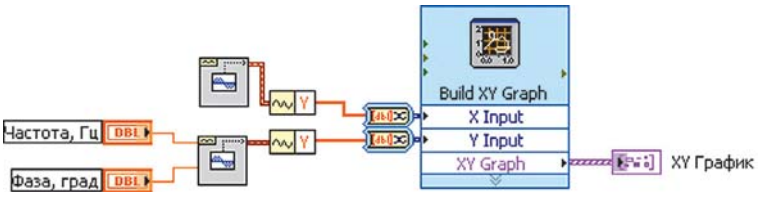
Итак, создадим и отобразим двухкоординатный график исходя из всего вышеописанного. На входы подадим гармонические сигналы, обеспечив возможность регулирования разности фаз между ними. Частота одного из колебаний будет постоянна (и равна 10 Гц), другого - переменная с возможностью ее управления (у реальных сигналов с клемм пассивного четырехполюсника частоты, естественно, совпадают). Как известно, получаемые кривые носят название фигур Лиссажу:



Данные из двух массивов типа Double Precision преобразуем в массив кластеров с помощью структуры ***For Loop*** (цикл с фиксированным количеством итераций):

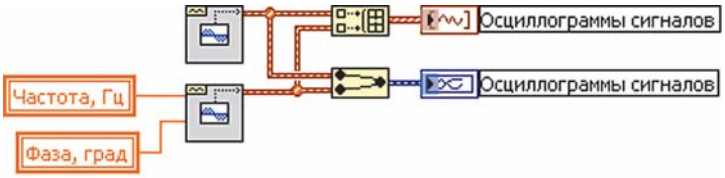


Для упрощения процедуры "сборки" типа данных ***XY Graph*** необходимо вызвать индикатор ***Express XY Graph***, находящийся в той же подпалитре, что и описанный выше ***XY Graph***. В этом случае вызова структуры ***For Loop*** удастся избежать - появившийся экспресс-VI сделает все самостоятельно:

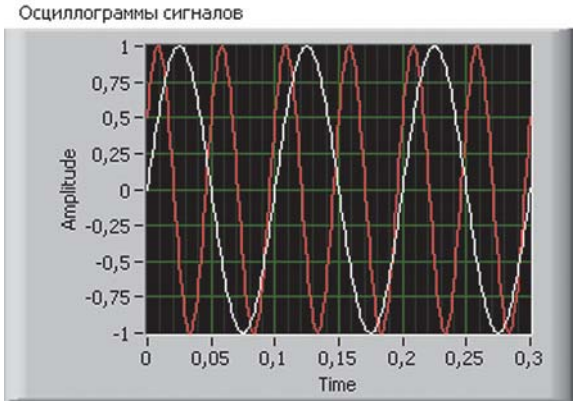


Тип входов экспресс-VI - ***Dynamic Data***. Однако беспокоиться о несовпадении типов не стоит - конверторы (функция ***Convert To Dynamic Data***, миниатюрные предвключенные иконки) устанавливаются автоматически.

Для контроля правильности построения фигур Лиссажу и дальнейшего их изучения полезно вывести данные с обоих входов на один график. Для этого необходимо использовать функцию ***Merge Signals*** (находится в подменю, расположенном по адресу ***Express»Signal Manipulation***), либо обыкновенную ***Build Array***. Эти функции позволяют "собрать" две последовательности данных воедино:



А для приведенной выше фигуры Лиссажу графическое представление данных будет следующим:



Как видим, можно отследить и отличие частот колебаний, и (при одинаковой частоте) - фазовый сдвиг между ними.

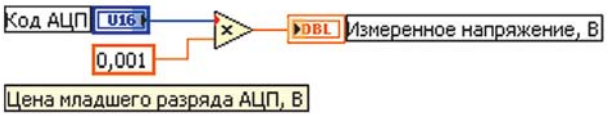
И, наконец, рассмотрим особенности обработки результатов, получаемых, например, от внешнего модуля сбора данных с интерфейсом USB, либо от встраиваемой платы АЦП, устанвленной в PCI-слот компьютера.

Основной особенностью обработки данных является то, что выходной сигнал АЦП является дискретным (т.е. кодом). Естественно, в вычислительной технике рассматривают именно двоичные коды, но созданная в LabVIEW система сбора данных, естественно, не может выдать результаты на экран в "первозданном" виде, иначе ее лицевая панель будет абсолютно "нечитабельной" для пользователя. Поэтому необходимо преобразовать получаемые коды в десятичную форму. Это делают либо непосредственно перед выводом численных данных, либо сразу после их получения от АЦП, еще перед математической обработкой в рамках приложения.

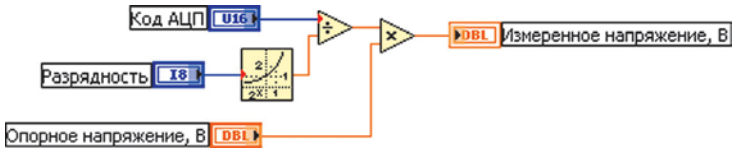
Последний вариант является более предпочтительным при написании приложений с относительно сложными алгоритмами цифровой обработки, либо при проведении косвенных измерений (когда используестя несколько каналов АЦП).

В первом случае Вы можете сильно упростить отладку Вашего приложения. Судите сами. Когда Вы используете инструмент ***Probe Data*** (напомним, он находится в палитре ***Tools Palette*** и весьма удобен для анализа работы частей блок-диаграммы "на ходу"), в получаемом окошке намного большую ясность дает число в десятичном представлении, чем последовательность из "0", "1".

Если Вы получаете результат преобразования в виде одного целого числа (функции чтения аналоговых данных большинства DLL-библиотек выдают результат именно такого типа), то поступаем следующим образом. Целое число представляет собой двоичный код, только в десятичном представлении. Т.е. каждый из двоичных разрядов уже был домножен на соответствующий весовой коэффициент и все произведения был сложены, а знак учитывался отдельно. Тогда Вам необходимо лишь знать цену (в Вольтах) младшего разряда АЦП, чтобы умножить на нее исходный код и получить требуемое напряжение:

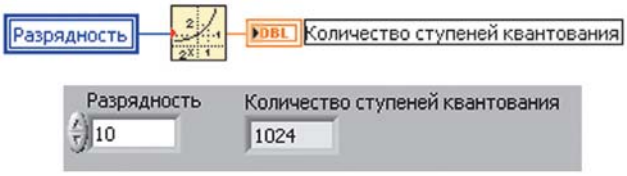


Можно также произвести расчет с использованием двух других величин - разрядности АЦП и значения его опорного напряжения. Эти данные приводятся в спецификациях преобразователей и модулей сбора данных в числе основных. Таким образом, цену младшего разряда,"ступени" АЦП Вы учтете автоматически, избегая погрешности вследствие ее округления:

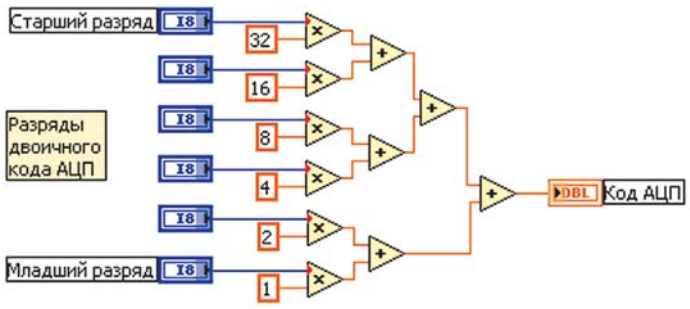


Очевидно, что функция ***Power Of 2*** в данном случае необходима для перехода от количества разрядов преобразователя АЦП к количеству ступеней квантования. Находится она в ***Mathematics»Elementary & Special Functions»Exponential***.

Если же используется представление цифровых данных в виде шины (массива или кластера переменных типа BOOL, либо целочисленных, но в диапазоне 0, 1), то необходима обработка каждой ее линии. Поскольку разрядность АЦП ясна исходя из количества этих линий (элементов массива, кластера), главное - знать порядок их размещения (т.е. какая из линий содержит старший разряд результата и т.д.) и опорное напряжение, либо цену ступени квантования АЦП.

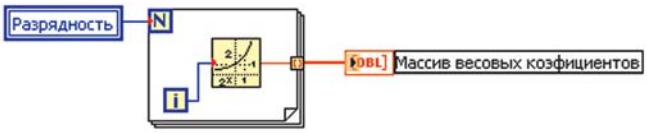


Весовые коэфициенты можно создать вручную, прописывая их в константах либо массиве/кластере констант:



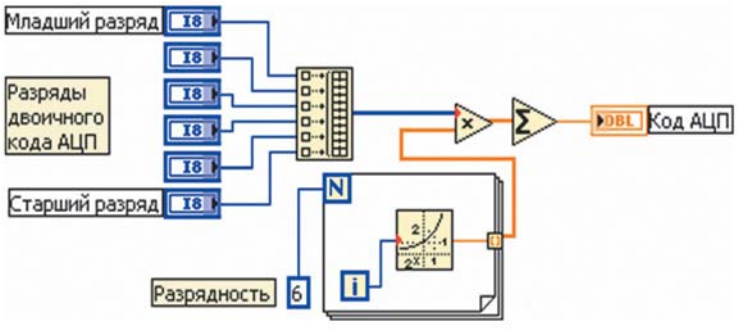
Здесь и далее как пример представлены варианты обработки шестиразрядного двоичного кода.

Возможен также более удобный вариант формирования массива весовых коэффициентов следующим образом:



Здесь также находит применение функция ***Power Of 2***. Именно такой закономерности подчиняются весовые коэфициенты. Кстати, все вышесказанное можно легко проверить путем использования ***Probe Data***.

Впоследствии достаточно выполнить попарное произведение элементов массивов и сложить полученные результаты:



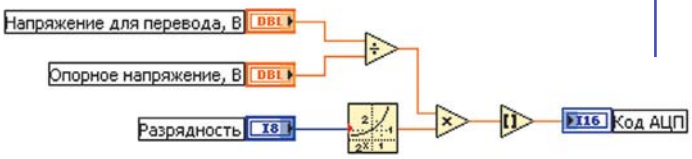
Далее необходимо осуществить уже описанное преобразование кода, заданного целочисленной переменной, в напряжение.

И, наконец, рассмотрим принципы преобразования "аналогового" сигнала (в нашем случае - типа "DBL") в

цифровой код для требуемых разрядности и опорного напряжения.

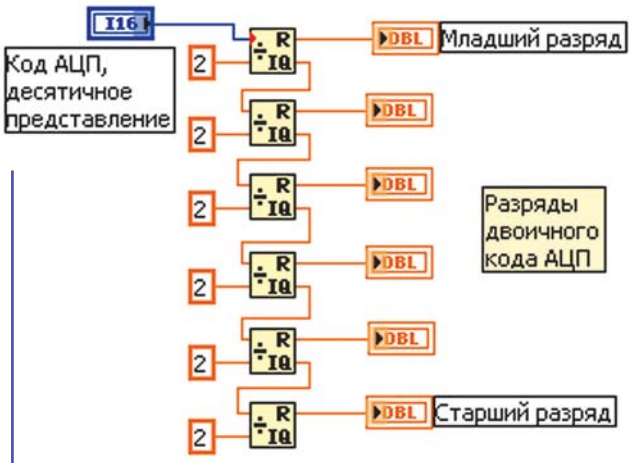
Как известно, перевод числа из десятичной формы в двоичную довольно просто осуществить, последовательно деля это число на основу требуемой системы счисления (т.е., на 2) и Вы уже вспомнили, с помощью какой функции проще всего выделить остаток от деления двух чисел? Естественно, эта функция - ***Quortier & Remainder*** (палитра ***Numeric***). Входы слева - делимое и делитель, справа - остаток и частное.

Так вот, сначала сформируем цифровой код в десятичном его представлении. Процедура обратна вышеописанному вычислению напряжения с использованием полученного от АЦП кода. Необходимо лишь следить, чтобы код был целым числом - т.е., округляя его после перевода.



А вот "получение" двоичного кода из соответствующего ему десятичного кода выполняем по стандартному математическому алгоритму: путем последовательного деления на 2 и "запоминания" остатков.

Например, перевод напряжения в шестиразрядный двоичный код (т.е. фактически моделирование работы шестиразрядного АЦП) со значением опорного напряжения 5 В можно выполнить следующим образом:



Как видно, уже нет необходимости создавать специальную последовательность вессовых коэффициентов, делить приходится постоянно на 2.

На сегодня пока все. В следующих выпусках практикума продолжим углубляться в LabVIEW и, конечно, будут ответы на актуальне вопросы, практические советы и интересные примеры.

Материал подготовлен студентами старших курсов Национального технического Университета Украины "Киевский политехнический институт" факультета авиакосмических систем, при технической поддержке ООО "ХОЛИТ Дейта Системс" (Киев).

e-mail: info@labview.com.ua