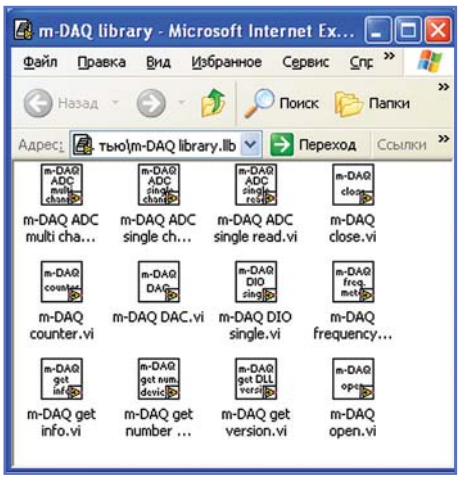
Учебный практикум по LabVIEW

Настоящим LabVIEW-программистом можно назвать того, кто умеет работать с реальным "железом". В простейшем случае это может быть или плата, встраиваемая в PC, или внешний модуль, сопрягаемый с компьютером по штатному интерфейсу. Хотите освоить технологии сбора данных в полном объеме – устройство ввода/вывода надо иметь в наличии. А порекомендовать для этих целей можно микросистему сбора данных m-DAQ (ХОЛИТ™ Дэйта Системс, Киев), доступную по цене даже для студентов

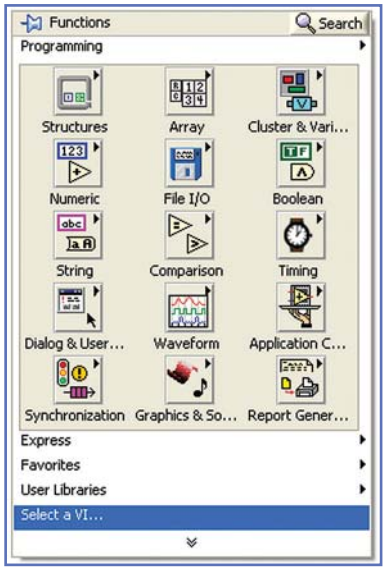
Все, что обычно присутствует в многофункциональных устройствах В/В сигналов для PC, а именно многоканальный АЦП, ЦАПы, каналы дискретного ввода/вывода и таймер-счетчик, в микросистеме m-DAQ с интерфейсом USB имеется. Аналоговый ввод содержит 8 каналов с возможностью индивидуального конфигурирования входного диапазона по напряжению (на этапе заказа). Разрядность АЦП 10 бит, а частота преобразования до 100кГц. Два канала аналогового вывода также могут быть сконфигурированы на требуемый выходной диапазон. Дискретный В/В содержит 10 линий в уровнях ТТЛ, причем каждый канал может быть индивидуально настроен на ввод или вывод. Кроме того, один из каналов дискретного В/В может использоваться как вход счетчика, а другой - как вход внешнего запуска АЦП или синхронизации. На контактах внешнего разъема присутствуют также напряжения питания +5В и ±15 В. Вот на примерах работы с такой микросистемой давайте и начнем осваивать работу с реальным "железом" в среде LabVIEW.

Средства программной поддержки m-DAQ включают DLL-библиотеку и примеры работы с ней. Ее можно использовать для работы в различных средах программирования, в т.ч. и в LabVIEW. Но, в большинстве случаев, уважающие себя производители в комплект поставки устройств сбора данных включают также llb-библиотеки для работы в LabVIEW. И m-DAQ не исключение. Библиотека подпрограмм ***m-DAQ library.llb*** - это то, что в данном случае нам нужно:

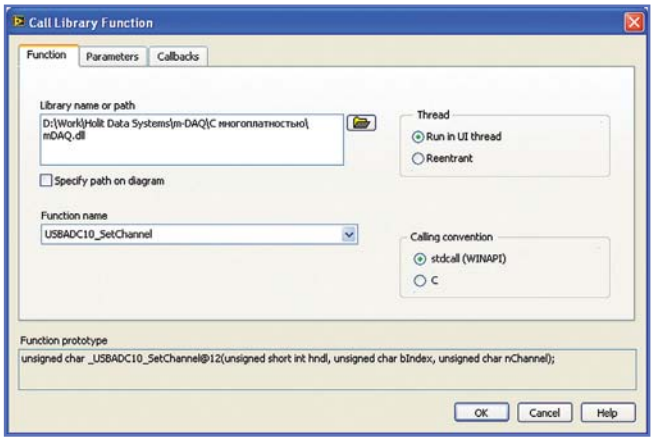


Уместно напомнить, что вызов subVI осуществляется путем выбора в меню ***Functions*** пункта ***Select a VI***... , без обращения к каким-либо подменю. В открывшемся после выбора Select a VI... окне необходимо указать путь к m-DAQ library.llb и требуемый в ней subVI.

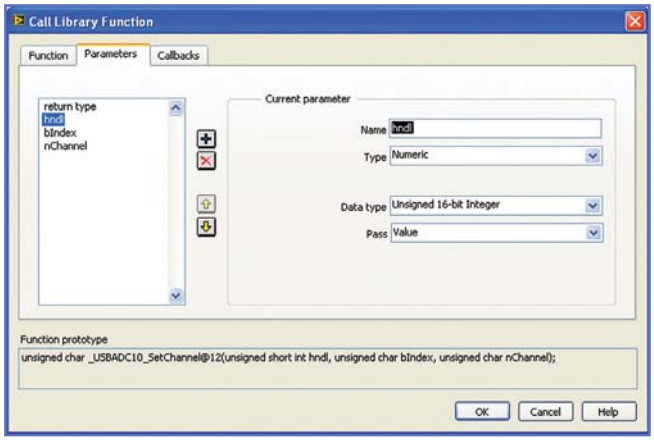
Для написания и работы программы без промедлений помещайте mDAQ.dll в одну папку с LabVIEW-библиотекой. В противном случае будет затрачено некоторое время на поиск DLL-библиотеки, и, при необходимости, будет выведено диалоговое окно для указания адреса mDAQ.dll вручную.



Вызов функций динамических библиотек в LabVIEW осуществляется с помощью функции ***Call Library Function Node***. Для его настройки необходимо произвести конфигурирование данного блока - указать путь к файлу библиотеки и перечень входных и выходных переменных с их типами.



Выполнение этих процедур - весьма кропотливое дело. Кроме того, допущенные при этом ошибки (например, установка типа вызова "C" вместо стандартного "stdcall (WINAPI)") могут вызвать не появление окна с предупреж-



дением и местом ошибки, а закрытие LabVIEW без каких-либо подсказок. А вот другие "вольности", например, несовпадение названий переменных при конфигурации в LabVIEW с названиями, приведенными в h-файле и руководстве пользователя или несовпадение размерностей массивов, не станут причиной ошибок. Поэтому начинающие пользователи могут испытывать определенные затруднения при работе с "чистой" DLL-библиотекой. DLL-библиотека содержит 18 функций, условно разделенных на 4 группы:

- функции общего назначения: получение информации об устройстве и DLL-библиотеке (для контроля соответствия их версий), открытие и закрытие устройства, получение специального указателя на устройство (переменной, необходимой для корректной адресации в случае подключения нескольких устройств к PC , однако указатель на устройство необходимо использовать и в случае работы с одним модулем);
- функции работы с АЦП (конфигурирование, однократное и многократное чтение) и ЦАП (однократный вывод данных);
- функции синхронизации (внешняя синхронизация АЦП либо его запуск по внешнему событию, запуск и останов счетчика импульсов);
- функции работы с каналами дискретного В/В (конфигурирование входов, однократные либо многократные чтение/запись).

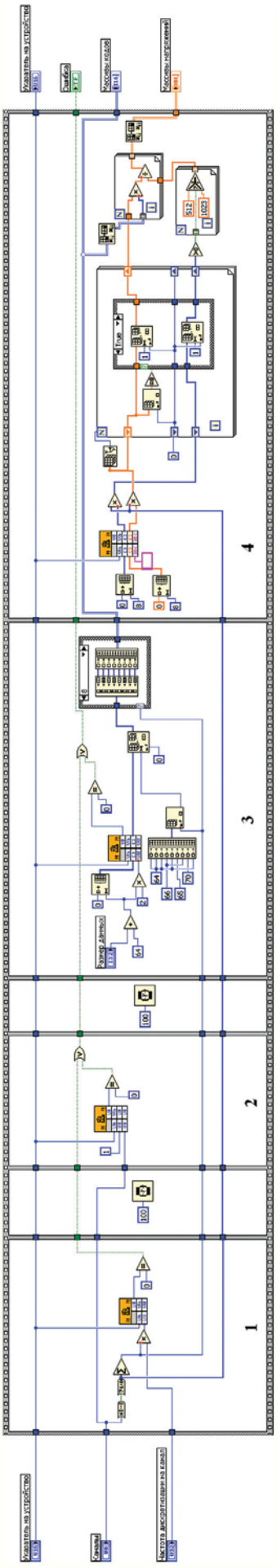
Каждая из 12-ти подпрограмм библиотеки m-DAQ library.llb позволяет выполнять одно выбранное простое действие, например, однократное чтение данных с АЦП, многократный опрос одного или нескольких его каналов, вывод данных на ЦАП, на линии дискретного В/В. Кроме того, подпрограммы m-DAQ library.llb можно использовать в качестве примера по управлению DLL-библиотекой при возникновении трудностей. Однако не следует думать, что подпрограммы полностью повторяют функции mDAQ.dll и состоят лишь из блоков ***Call Library Function Node***. Более того, в разных подпрограммах может быть использован вызов одинаковых DLL-функций, лишь с разной конфигурацией и последующей обработкой данных. Все дело в том, что функции, выполняемые с помощью подпрограмм, являются несколько "высшим уровнем" по отношению к непосредственно DLL-функциям.

Для примера приведем блок-диаграмму подпрограммы последовательного опроса нескольких каналов АЦП ***m-DAQ ADC multi channel.vi*** с некоторыми пояснениями.

Программный код состоит из последовательности фреймов ***Flat Sequence***, в каждом из которых помещен либо один блок ***Call Library Function Node*** с функциями сопутствующей обработки данных, либо ***Wait (ms)***, предназначенный для кратковременной задержки между обращениями к модулю. Задержка является "профилактическим" действием. Следует заметить, что от момента обращения к модулю до непосредственного выполнения действий в любом случае существует некоторая временная задержка. Если частота измеряемых сигналов или исследуемых событий весьма высокая, задержки могут отрицательно повлиять на результаты измерения, вплоть до пропуска информативных участков измеряемых сигналов. Точные значения задержек привести весьма трудно, однако можно установить экспериментально. Специально для минимизации этих паразитных явлений (наряду с возможностью запуска АЦП в режиме ожидания события либо внешнего тактирования АЦП) существует DLL-функция ***USBADC10_SetExternalStart***.

На блок-диаграмме ***m-DAQ ADC multi channel.vi***, представлены следующие фрагменты:

- Установка с помощью библиотечной функции ***USBADC10_Set FreqADC*** требуемой частоты дискретизации для последующего ввода данных. Этой установке предшествует перерасчет заданной для одного канала частоты дискретизации (соответствующий вход подпрограммы, терминал представлен слева), т.е. умножение ее значения на количество каналов. Количество каналов явно не задано, поскольку переменная "Каналы" содержит байт соответствующих флагов,

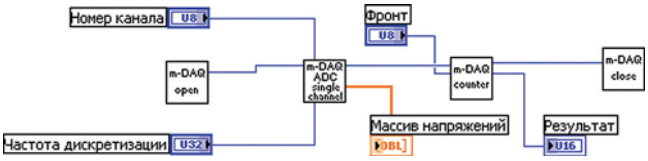


где каждый его бит указывает, требуется ли опрашивать канал. Например, при установке "Каналы" = 3_{10} = 00000011_2 будет произведен поочередный опрос 0-го и 1-го каналов АЦП).

Для расчета количества каналов используются функции ***Number To Boolean Array, Boolean To (0,1)*** (на ее выходе получаем массив из 8-ми целочисленных элементов 0/1, указывающих, какие каналы необходимо включить; массив будет использован в дальнейшем), и ***Add Array Elements***.

Для контроля безошибочного выполнения функции ***USBADC10_SetFreqADC*** используется ее выход. Впоследствии выходы всех ***Call Library Function Node*** объединяются и результат подается на выход subVI "Ошибка". Подпрограмма считается выполненной с ошибкой, если выполнение хотя бы одной ***Call Library Function Node*** ошибочно.

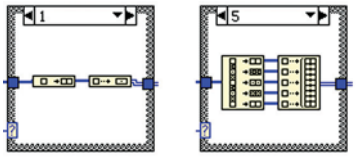
Вход подпрограммы "Указатель на устройство" предназначен для указания, к какому из модулей, подключенных к PC, следует обратиться. Этот вход дублируется идентичным выходом, представленным в правой части блок-диаграммы. Дублирование сделано для избежания необходимости использования фреймов ***Flat Sequence*** при работе с уже готовыми подпрограммами библиотеки. Как Вы, вероятно, знаете, ***Flat Sequence*** -весьма неудобный инструмент для установки порядка выполнения составляющих блок-диаграммы. Очень многие LabVIEW-программисты считают, что от фреймовых структур нужно полностью отказаться в пользу простых терминалов, подобных кластерам ошибок (например, в том же Call Library Function Node), ибо модификация фреймов в уже написанной программе может быть рутинным и затруднительным делом. Ниже представлен пример программного кода с последовательным вызовом подпрограмм работы с m-DAQ. Проводник, содержащий указатель на устройство, последовательно соединяет иконки subVI. Очевидно, что при необходимости модификации, например, изменения последовательности выполнения подпрограмм, необходимо "разорвать" этот проводник и поменять последовательность подключения иконок. Если бы Вы имели дело с ***Flat Sequence***, изменение последовательности выполнения было бы сделать труднее.



■ Установка с помощью функции ***USBADC10_SetChannel*** номеров каналов, которые необходимо опросить. Константа "1", подключенная ко входу ***bIndex*** указывает, что следует провести именно многоканальный опрос.

Независимо от установленных в ***USBADC10_SetChannel*** значений флагов каналов опрос происходит в порядке возрастания номера канала. Это следует учитывать при обработке полученных с АЦП данных. В OEM-версиях m-DAQ АЦП имеет шесть каналов. Поэтому во избежание ошибки вызова функции, следует корректно задавать параметры этой функции.

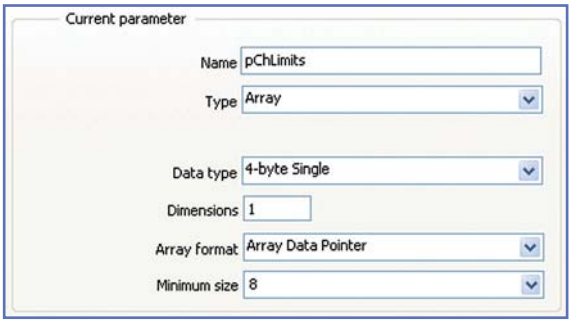
■ Запуск режима чтения АЦП с помощью функции ***USBADC10_BulkReadDataADC*** и получение массива кодов. После получения данных проведено удаление первых 64-х отсчетов (использована ***Delete From Array***) для исключения фиксации переходных процессов. После этих процедур выполняется разделение массива кодов на несколько массивов (точнее, матрицу, строки которой являются массивами данных с каналов) с помощью ***Decimate 1D Array***. Использование именно этой функции обусловлено характером размещения данных в массиве с выхода ***USBADC10_BulkReadDataADC*** (они "перемежаются"). Количество массивов равняется количеству опрошенных каналов, а применение для этого ***Case Structure*** обусловлено различным количеством каналов.



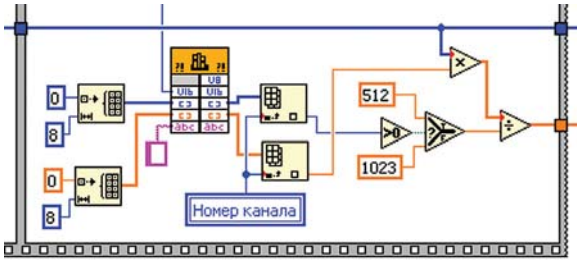
■ Перевод полученных значений кодов в значения напряжения. Для этого выполняется вызов функции ***USBADC10_GetDeviceInfo*** и получение информации о модуле, а именно массива типов входов АЦП. Каждый элемент массива - 0 в случае, если вход однополярный, либо 1 в противном случае. Диапазон кодов до программной калибровки для однополярных каналов составляет 0..1023, а для биполярных -511..+512.

Программная калибровка выполняется внутри модуля автоматически с целью повышения точности АЦП и состоит в следующем. "Сырой" код складывается с записанной в памяти модуля константой, после чего - перемножается еще с одной константой.

Как известно входы ***Call Library Function Node*** дублируются одноименными выходами и наоборот, за исключением "главного" выхода ***return type***, расположенного над всеми информационными выходами. Если выход содержит обычную переменную (не указатель на нее), получить ее значение можно, просто "вытянув" проводник с выхода. Если же в перечне параметров переменной DLL-функции представлен указатель (что выбрано в поле ***Array format, String format, Pass***), то необходимо подключение к соответствующему входу переменной либо константы соответствующего типа, даже если данные с выхода не будут использованы.



На диаграмме, представленной ниже, показано, как "зарезервировать" два массива для параметров каналов АЦП с помощью функции ***Initialize array*** и подать созданные массивы на входы Call Library Function Node, обращающегося к USBADC10_GetDeviceInfo. К одному из входов подключена константа строкового типа - "резерв" под серийный номер, который в данном случае не использован.



Сначала, с помощью ***For Loop***, происходит "чистка" массивов типов входов и их предельных напряжений - информация о неиспользуемых входах удаляется. Для проверки и последующей "чистки" используется массив, полученный во фрейме №1. После всего этого проводятся простейшие арифметические операции для перехода от кода к напряжению. Каждая строка матрицы полученных кодов делится на значение предельного напряжения соответствующего канала и умножается на 512 либо 1023 в зависимости от типа последнего.

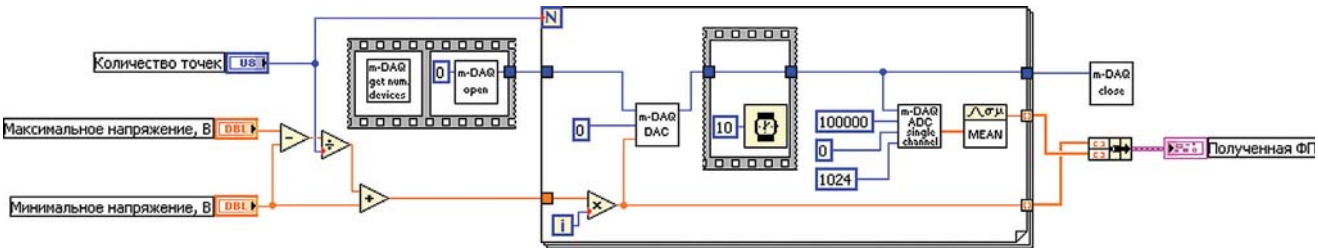
Последовательность действий для создания приложения по управлению m-DAQ следующая. Сначала необходимо обратиться к функции получения количества подключенных устройств ***USBADC10_GetNumberDevices*** либо вызвать соответствующую подпрограмму ***m-DAQ get number devices.vi***. Данная функция создает указатели на каждый из подключенных в момент ее вызова модулей, без которых последующая работа невозможна. При подключении нового m-DAQ после вызова этой функции корректная работа с ним и подключенными ранее модулями не гарантируется. Далее, непосредственно перед работой (снятием данных с АЦП и т.п.) необходимо произвести "открытие" m-DAQ с помощью ***USBADC10_OpenDevice*** или ***m-DAQ open.vi***, выходной параметр которой - рассмотренный ранее указатель на открытое с ее помощью устройство.

Указание, какой именно из модулей необходимо открыть, делается при вызове ***USBADC10_OpenDevice***. Для этого указывается номер требуемого m-DAQ, в порядке его подключения к шине USB. Нумерация начинается, естественно, с 0.

И, наконец-то, рассмотрим простейший пример использования микросистемы. Для этого реализуем выполнение следующей задачи - получение функции преобразования (ФП) четырехполюсника.

Подключая исследуемый объект к m-DAQ, необходимо следить за правильностью соединений входов и выходов $A_{IN}0$ и $A_{OUT}0$. И не забудьте общий провод четырехполюсника подключить к линии A_{GND}, иначе результаты будут получены некорректными. А еще всегда следует помнить, что неправильное подключение источников и приемников сигналов способно, мягко выражаясь, перевести m-DAQ в нерабочее состояние.

Блок-диаграмма программы для получения ФП будет выглядеть так:



Входными переменными являются предельные значения входного напряжения и количество точек получаемой ФП. Как видим, переход от предыдущей точки ФП к следующей организован с помощью цикла ***For Loop***. В теле цикла происходит подача некоторого рассчитанного напряжения на ЦАП (это напряжение является для четырехполюсника входным), снятия массива напряжений с 0-го входа АПЦ и расчет среднего значения этого массива.

Для устранения влияния переходных процессов из-за влияния реактивностей четырехполюсника установим после ***m-DAQ DAC.vi*** блок ***Wait (ms)***, обеспечивающий "простой" выполнения в течение 10 мс.

Поскольку при выполнении рассматриваемого примера работа производится только с постоянными сигналами, для получения большей точности вычисления ФП во время каждого отдельного измерения производится получение не одного значения напряжения с выхода четырехполюсника, а выборка из 1024-х таких отсчетов. Как следует из блок-диаграммы, частота дискретизации установлена равной 100 кГц. Впоследствии с помощью функции ***Mean*** рассчитывается среднее арифметическое данного массива, что позволяет избавиться случайных погрешностей, обусловленных шумами оборудования. Далее (после ***For Loop***) массивы входного и выходного напряжений с помощью функции ***Bundle*** объединяются в кластер для удобства последующего использования и графического представления результата работы.

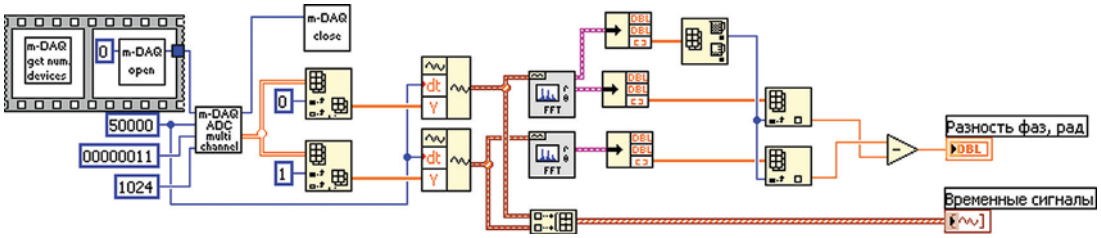
Повысить точность ФП возможно также при использовании еще одного канала АЦП, например $A_{IN}1$, производя с его помощью оцифровку входного сигнала четырехполюсника, исключая таким образом, в некоторой мере, погрешность ЦАП.

В порядке упражнения модифицируйте программу так, чтобы выполнять получение ФП нескольких четырехполюсников одновременно. Для этого будет достаточно вынести во входные переменные главной программы соответствующие входы ***m-DAQ DAC.vi*** и ***m-DAQ ADC single channel.vi***, которые на рассмотренной диаграмме присоединены к константам "0".

Если кому-то захочется снять частотные характеристики таким же образом, то следует помнить, что ЦАПы в m-DAQ построены по ШИМ-методу, т.е. предназначены для формирования сигналов практически постоянного напряжения. Но, если все-таки очень хочется, тогда надо либо использовать внешний генератор синусоидальных сигналов, управляемый линиями дискретного В/В m-DAQ, либо использовать более производительную систему сбора данных с быстродействующими ЦАП. Причем во втором случае и для снятия АЧХ, и для снятия ФЧХ, желательно задействовать еще один вход АЦП.

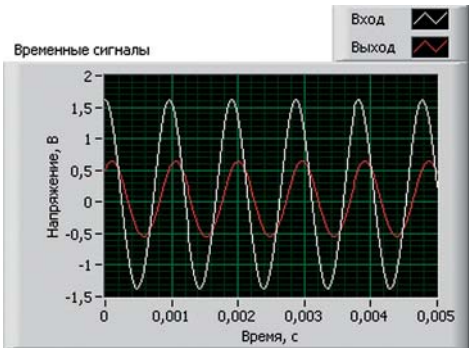
Рассмотрим еще один несложный пример - расчет разности фаз между выходным и входным сигналами четырехполюсника. В приведенной программе данные поступают с каналов АЦП $A_{IN}0$ и $A_{IN}1$. При этом используется подпрограмма ***m-DAQ ADC multi channel.vi***. Поскольку для дальнейших вычислений требуется одновременное получение сигналов с обоих входов АЦП, использование ***m-DAQ ADC single channel.vi*** в этой ситуации уже не допускается. Для уменьшения погрешности несинхронности взятия отсчетов для аналоговых входов, частоту дискретизации устанавливаем максимально возможную, т.е. 50 кГц на канал. Константа 00000011_2 используется для задания рабочих каналов (первых двух).

После выполнения ***m-DAQ ADC multi channel.vi*** получаем матрицу результатов размером 2 строки х 1024 столбца.



Формируем из нее две осциллограммы типа ***Waveform***. И после этого вычисляем с помощью FFT Spectrum (Mag-Phase) амплитудные и фазовые спектры сигналов с аналоговых входов. Фазы гармонических колебаний получаем, извлекая из массивов фазовых спектров отсчеты, соответствующие максимуму амплитудного спектра. Таким образом, в этой программе вычисляется разность фаз независимо от частоты сигналов (естественно, в пределах ограничений, регламентируемых теоремой Котельникова-Найквиста).

Пример реальных осциллограмм сигналов, полученный с помощью описанной программы, представлен ниже. На графике отчетливо видны понижение уровня сигнала и фазовый сдвиг.



Повышение точности возможно за счет устранения погрешности от несинхронности работы аналоговых входов (ненулевого времени переключения между ними). Это следует реализовать либо путем теоретического расчета временной задержки по известной частоте дискретизации, либо путем экспериментального получения этой задержки. Во втором случае необходимо подать один сигнал известной частоты одновременно на два аналоговых входа и рассчитать разность фаз с помощью вышеприведенной программы. А далее, от полученной разности фаз нетрудно перейти к значению временной задержки.

Для закрепления материала текущего практикума, рекомендуется создать ряд виртуальных приборов - "осциллограф", "анализатор спектра", "частотомер" и "вольтметр". Кстати, в комплекте поставляемого с m-DAQ ПО, такие приборы есть, но исходных текстов программ нет. Попробуйте сделать что-то подобное. Удачи!

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

e-mail: info@labview.com.ua