Уроки по LabVIEW

Основные элементы графического программирования освоены Вами в полном объеме. Уроки базового курса как бы позади, но тема не исчерпана. LabVIEW вообще неисчерпаем, поскольку это не просто программный продукт, LabVIEW – это профессия. Поэтому материалы последующих выпусков весьма условно можно считать уроками, скорее это очередные темы для освоения, которые будут полезны для решения круга задач, не рассматривавшихся ранее.

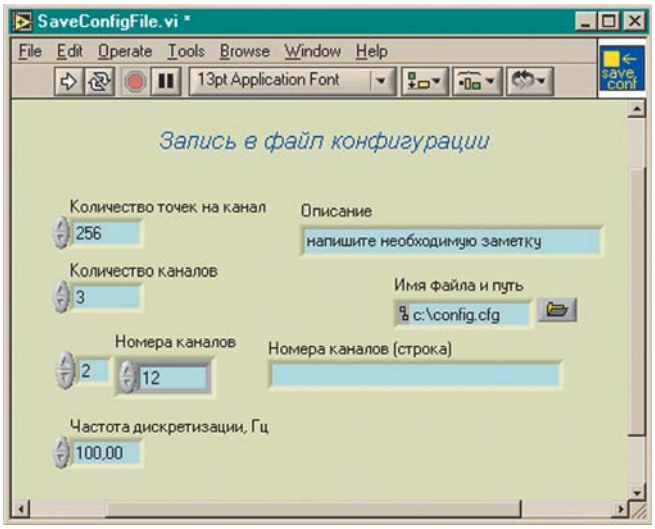
На этом уроке Вы научитесь создавать и использовать файлы конфигурации, подпрограммы для записи и чтения данных в/из файла конфигурации, а также обрабатывать события, связанные, например, с перемещением манипулятора.

Работая с платами расширения, очень часто приходится выдумывать свой собственный формат данных для записи или чтения параметров конфигурации. Причем фрагменты программ или подпрограммы получаются достаточно громоздкими и нечитаемыми. Приходится делать анализ файла, создавать обработчик строковых и числовых выражений и т.д. Чтобы избежать всей этой кропотливой работы по созданию и отладке таких подпрограмм, в LabVIEW реализован очень удобный механизм для работы с файлами конфигурации. Набор виртуальных инструментов (подпрограмм) находится в функциональной панели: ***Functions » File I/O » Configuration File Vis***.

Давайте создадим библиотеку, в которую поместим две подпрограммы по работе с файлами конфигурации. Первая подпрограмма будет формировать такой файл и обеспечивать запись в него необходимой информации, а вторая будет использоваться для чтения параметров из него.

Представим себе, что мы работаем с платой многоканального АЦП и хотим создать файл отчета, который будет содержать интересующие нас параметры: количество каналов аналогового ввода, номера каналов, частоту дискретизации, количество регистрируемых отсчетов или интервал времени наблюдения, рабочие диапазоны или коэффициенты усиления, и просто текстовые замечания, а может быть что-то еще.

Создайте новый VI с именем SaveConfigFile и установите элементы управления и индикации на интерфейсную панель как это показано ниже. Как это сделать, объяснять Вам не нужно. Ведь базовый курс уроков по LabVIEW позади. Работы всего максимум на три минуты.

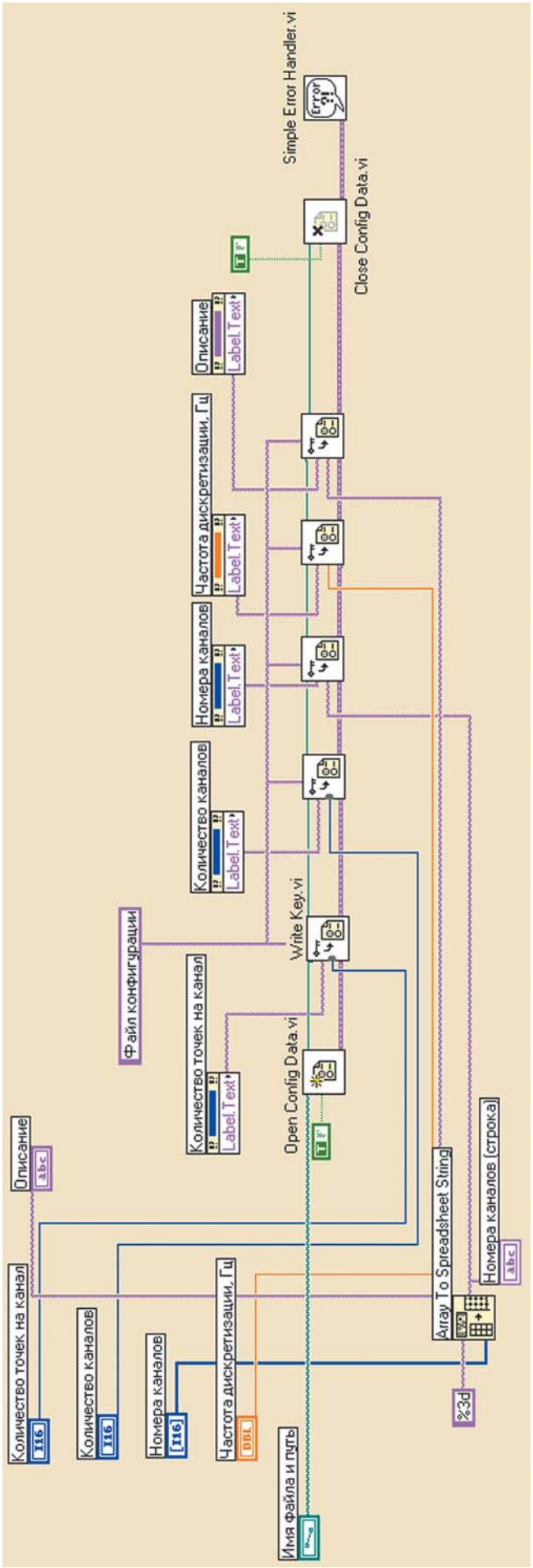


А теперь сделаем функциональную "начинку" программы. Для записи данных в файл любого формата, как Вы уже знаете, необходимо сначала открыть доступ к файлу, затем записать данные и наконец, "закрыть" работу с файлом. Файлы конфигурации не являются исключением. Следует обратить внимание на то, что массив чисел в рассматриваемом примере сначала преобразовывается в строку, а уж потом только записывается. Чтобы не создавать дополнительных строковых констант или элементов управления, для заголовков следует использовать свойство ***Label Text***. Причем сделать это нужно для каждого элемента, параметр которого Вы хотите сохранить в файле конфигурации. Для записи параметра используйте функцию ***Write Key.vi***. Выполните терминальные соединения как это показано ниже на диаграмме и можно считать, что работа сделана. Почти! Это ведь подпрограмма и Вы ее будете использовать в своих приложениях. Не забудьте нарисовать иконку.

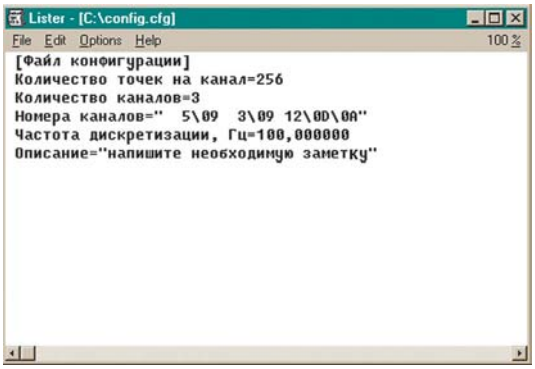
Несмотря на кажущуюся на первый взгляд сложность графической диаграммы, выглядит она абсолютно читаемой и простой для понимания и создания аналогичного алгоритма, но уже для Ваших задач. Можно сказать, что алгоритм получается "линейным", т.е. не содержащим никаких циклов и условных переходов.

Следует отметить, что данные сохраняются в общепринятом текстовом формате. Введите необходимые значения и пути и запустите программу на выполнение.

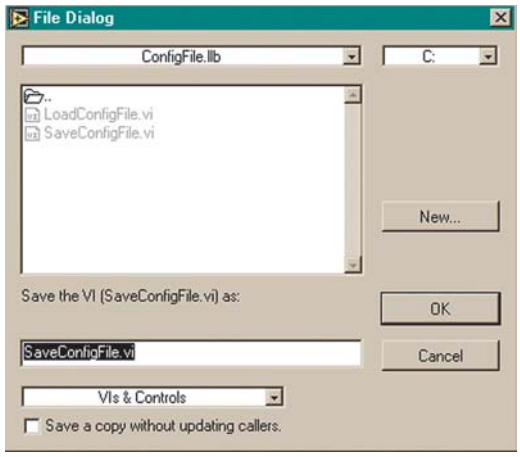
В результате работы Вашей программы будет создан



текстовый файл, который можно просмотреть с помощью любого текстового редактора:

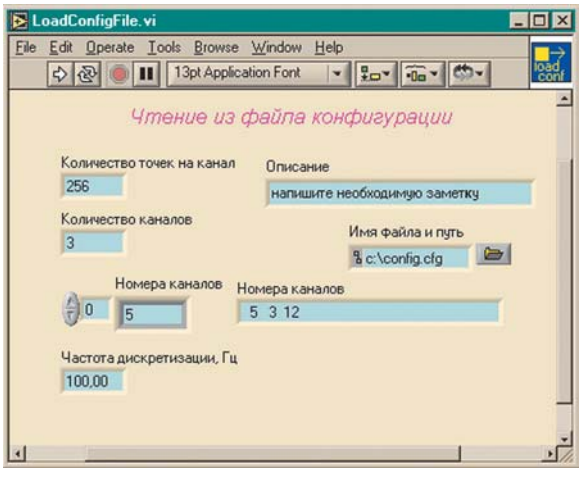


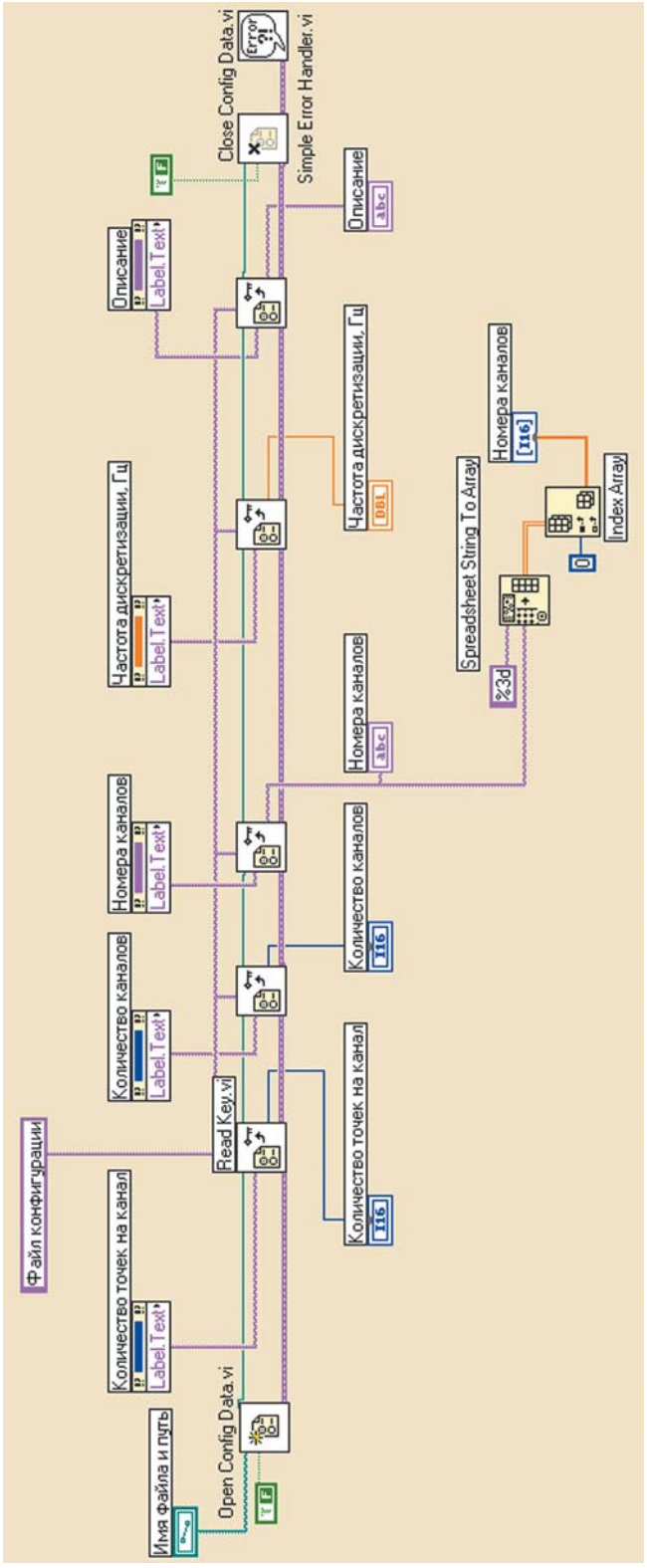
Следующий шаг - создание библиотеки, куда и следует поместить подпрограммы для работы с файлами конфигурации:



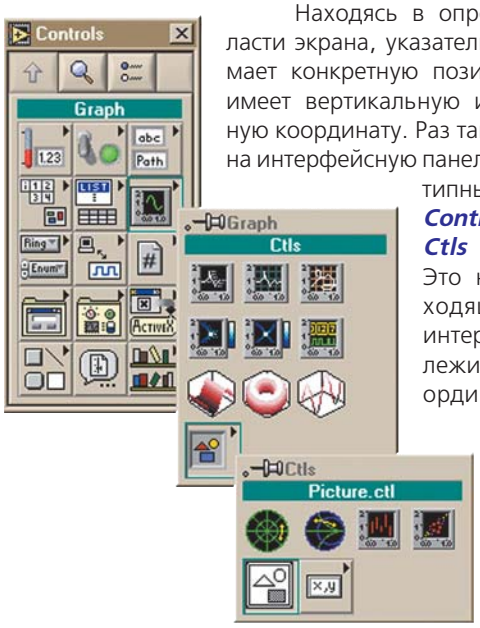
По аналогии с предыдущей подпрограммой, напишите программу для чтения файла конфигурации. Только при этом вместо функции записи ***Write Key.vi*** используйте функцию чтения ***Read Key.vi***. В результате у Вас должно получиться что-то похожее на представленные ниже интерфейсную панель и графическую диаграмму. И у этой программы "исходный текст" получился простым. Действительно, механизм взаимодействия с файлами конфигурации в LabVIEW реализован очень удобно. Он позволяет избежать многих проблем и сократить сроки написания программ. Не забывайте об этом.

Вводим путь, где находится файл конфигурации и запускаем программу на выполнение. Задача решена:

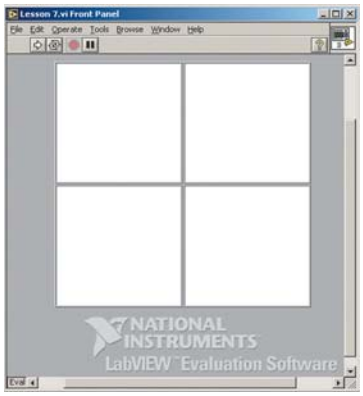




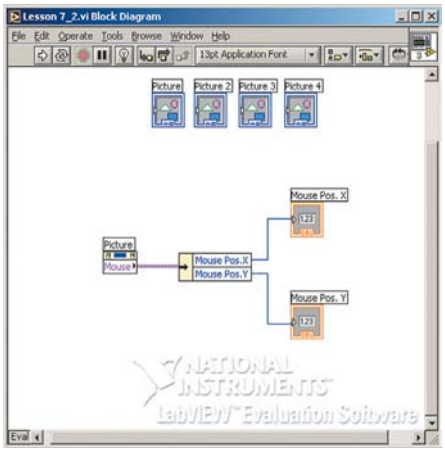
Переходим ко второй части урока. Такие действия, как нажатие пользователем клавиш клавиатуры, “кликанье” кнопкой мышки или ее перемещение генерируют события. Эти события передаются компоненту, в котором произошло событие. При каждом воздействии на аппаратуру генерируется свое событие. События могут быть очень детализированными. Например, событие для клавиатуры генерируется при нажатии и отпускании клавиши. Событие для клавиатуры генерируется также, когда на клавиатуре происходит автоматический повтор. Но давайте не будем усложнять картину, а лучше продемонстрируем обработку событий на примере манипулятора "мышь".



Находясь в определенной области экрана, указатель мышки занимает конкретную позицию, а значит имеет вертикальную и горизонтальную координату. Раз так, то установим на интерфейсную панель четыре однотипных компоненты: ***Controls » Graph » Ctls » Picture.ctl***. Это наиболее подходящий элемент интерфейса для отслеживания двухкоординатного манипулятора. В результате получаем четыре пустых белых квадрата. Для первого из них (верхний левый) создадим свойство Mouse, а из него с помощью функции для работы с кластерами ***Unbundle By Name***, выделим горизонтальную и вертикальную координаты положения мышки для текущего элемента. Установим два элемента индикации для полученных координат.



Запускайте программу на выполнение в циклическом режиме. Что Вы видите, манипулируя "мышью"? Если указатель манипулятора находится внутри объекта, то его координаты не нулевые, а принимают значения от нуля до максимума, в зависимости от размеров объекта. Попробуйте изменить размер объекта Если же указатель находится вне области объекта, то возвращается значение равное -1.

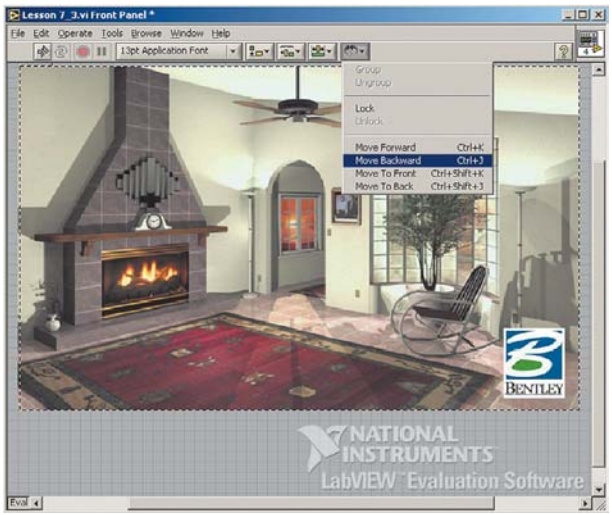


Теперь сделаем интересный и хитрый ход. Хотя первый хитрый ход уже сделан. Те из Вас, кто не просто читает эти заметки, а сидит у компьютера, должен был обратить внимание на то, что приведенная диаграмма по внешнему виду несколько отличается от того, что Вы видите на экране своего монитора. Почему?

Весь мир уже год, как перешел на новую версию LabVIEW. И Вам пора. Со следующего урока работаем в LabVIEW 7.1!

Но продолжим. Сделайте элемент и его рамку невидимыми. Для этого нужно выбрать "невидимый" цвет закраски. Запускайте программу на выполнение и поводите указателем манипулятора внутри невидимой области. Координаты все-таки возвращаются несмотря на то, что область прозрачная. Ну и что? А то, что поместив такой невидимый объект, например, поверх картинки, Вы сможете отслеживать положение указателя "мыши" как бы для рисунка, а не для заданной области.

Давайте создадим новое приложение, в котором будет только один элемент ***Picture***. В первую очередь, поместите рисунок на интерфейсную панель. Сделайте его фоновым. Для этого в выпадающем меню выберите поочередно ***Move Backward и Move To Back:***



Далее проделайте операции, которые Вы выполняли в предыдущем примере. Установите элемент ***Picture.ctl*** на интерфейсную панель, измените размер до размера картинки и сделайте его невидимым.

Теперь Вам предстоит "написать" фрагмент программы, который будет называть предмет интерьера комнаты, когда указатель "мышки" находится на нем, и, кроме того, при нажатии левой клавиши "мышки" фрагмент интерьера будет выделяться полупрозрачным фоном. Для этого просто необходимо проанализировать текущие координаты положения указателя манипулятора. Константы, которые используются для сравнения, подбираются экспериментальным путем. Для этих целей как раз и следует использовать индикаторы положения указателя. Это необходимо во время отладки программы. Но после ее завершения можно их и удалить.

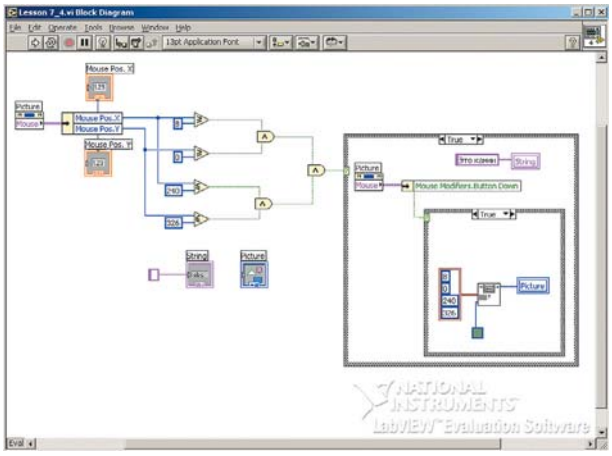
Составьте диаграмму как показано ниже. Следует обратить внимание на то, что в программе используются две структуры выбора. Причем в обоих случаях при значениях ***False*** области будут пустыми.

Обработка события реализована только для одного элемента интерьера - камина. Вы можете проделать самостоятельно действия для выделения остальных элементов, например, двери, кресла или окна. Структура будет абсолютно аналогичной, только координаты будут разными. Особое внимание следует обратить на то, что для "подсветки" выбранного объекта используется функция

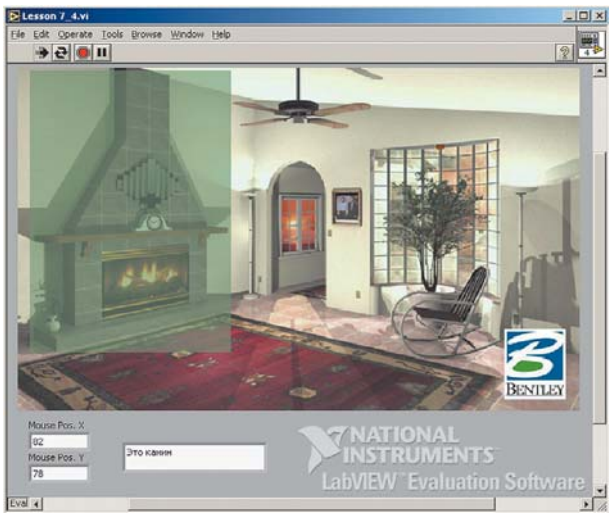
Draw Grayed Out Rect.vi, которая находится в ***Functions » Graphics & Sound » Picture functions » Draw Grayed Out Rect.vi***.

В рассматриваемом примере используется рисунок с интерьером, но это может быть не обязательно комната, а например мнемосхема технологического процесса. И подводя указатель манипулятора к необходимому объекту, можно реализовать более сложную структуру, например, вызов подпрограммы и т.п.

Какую диаграмму в результате Вы должны получить:



А в работе программа будет выглядеть так:



Здорово, не правда ли? Наверное многим из Вас уже хочется, чтобы компьютер голосом называл предметы. Вы чувствуете, что с LabVIEW это можно сделать очень просто? Действительно можно и действительно просто. Но "не следует бежать впереди паровоза". :-) Лучше подготовьтесь к работе в новой версии LabVIEW.