Уроки по LabVIEW

– ...
– Речь о паровозе на прошлом уроке шла?
– ... Да
– Так, почему возникли проблемы? Еще "вчера" программирование для многих из Вас было чем-то очень далеким и недоступным. А пройдя заочно базовый курс LabVIEW не все, но некоторые, возомнили себя фигурой, равной, сами знаете кому... Конечно, нужно "ковырять" это LabVIEW, все в уроках не рассмотреть. Но прежде чем браться за серьезный проект, следует усвоить ряд правил. И в этой связи весьма уместно прислушаться к рекомендациям опытного LabVIEW-программиста г-на Rande Johnson (Stress Engineering Services). Всегда следуйте его советам в любом проекте, большом или маленьком, простом или сложном...

1 Проект сначала, код потом

Наверняка этот совет Вы слышали неоднократно прежде, но дело все в том, что это - очень хорошее правило, можно сказать даже правило №1. Сядьте с несколькими листами бумаги и запишите ВСЕ, что Ваш заказчик требует от программы, нарисуйте, как должна выглядеть передняя панель, интерфейс... Думайте, уточняйте и при этом обязательно фиксируйте мысли и сказанное на бумаге. По сути, Вы создаете Программный Документ Требований *(по-нашему техническое задание на программное обеспечение, или кратко - ТЗ)*. Следует осознать, чего Вы собственно, в конечном итоге, хотите достигнуть.

На этапе формирования ТЗ LabVIEW может быть использован как быстрый и удобный инструмент для макетирования. Совсем не обязательно, чтобы передние панели работали или имели законченные диаграммы. Прототипы передних панелей помогут Вам далее, при написании части программ интерфейса пользователя. Затем задумайтесь о структурах данных, потоке программы, событиях, диаграммах. Мысль должна работать в терминах концепций потока данных, ведь LabVIEW основан на этом. И, наконец, учитывайте время реализации проекта и его бюджет. Если Вы делаете кое-что быстрое и черновое, приемлемо упростить этот процесс, чтобы скорее начать программировать. А если же Вы создаете ПО, которое будет необходимо модернизировать и для которого необходимо выполнить большое число дизайн-проектов передних панелей, следует спешить "постепенно" - версия 2, версия 3 и т.д.

2 Создавайте и проверяйте фрагменты программы, прежде чем включать их в основной алгоритм

Представьте себе человека, 10 лет программирующего в LabVIEW, который так и не смог запомнить, как использовать ***Threshold 1D Array*** или ***Interpolate 1D Array***. Бывает и такое, ничего страшного. Быстренько создается маленький VI для тестирования, чтобы найти правильный вариант перед включением этого фрагмента в блок-схему. Следует удостовериться, что этот фрагмент программы будет работать. Поверьте, что это правило предохранит Вас в дальнейшем от многих неприятностей.

3 Планирование многократного использования фрагментов программы

Вы написали часть программы - сохраните ее! Возможно, она будет востребована Вами если не в текущем проекте, то в следующем, а кроме того, этот фрагмент может быть и примером для Ваших коллег. Не сохраняя испытанные VI Вам придется потратить много времени на их восстановление. Гигабайты ныне подешевели, поэтому не скупитесь и не ленитесь. *Помните, что скупой два раза платит, а дурной два раза делает!*

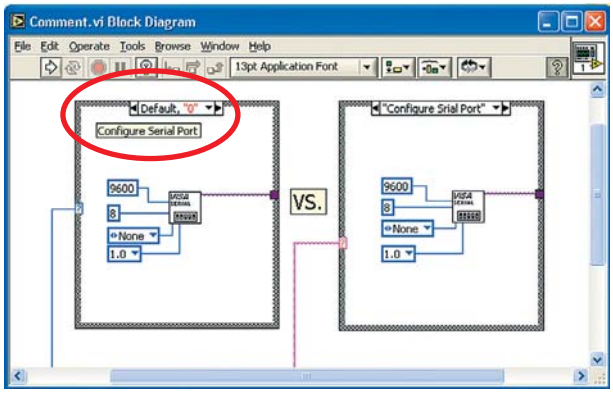
4 Делать все нужно очень аккуратно

Исходный текст, т.е. диаграмма, должна быть "читабельна". Да, для этого потребуется время. Но еще больше времени уйдет на то, чтобы разобраться даже в своих "каракулях". Приучив себя к четкости, чистоте и порядку в отношении диаграмм, Вы заметите, что и ошибок, и промахов станете делать существенно меньше.

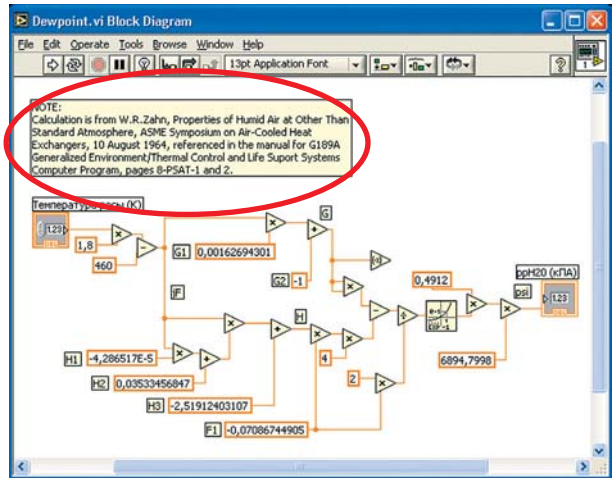
5 Комментарий, комментарий...

LabVIEW-программисты - народ занятой. Но комментарии по ходу программы просто НЕЗАМЕНИМЫ. Комментируйте все: хитрый код, жесткие алгоритмы, заказные утилиты и т.п. При этом комментарии не должны повторять диаграмму, хороший комментарий разъясняет Ваши намерения. Формулируйте кратко, обычно не более, чем одно или два предложения, описание каждого VI в его VI-информационных экранах.

Маркируйте все фрагменты диаграммы, которые, по Вашему мнению, неочевидны. Как минимум фиксируйте все структуры (циклы с условием продолжения для циклов, событий и последовательностей). Очень удобные штучки в LabVIEW - ***strbng*** и ***enumerated cases***. Такие структуры обязательно должны содержать пояснения:



А еще следует включать любые ссылки на источники информации, которые Вы используете. Например, в Вашей программе есть фрагмент, который преобразовывает значение точки росы ***dewpoint*** к соответствующему значению давления воды по достаточно непростой формуле. Если Вы нашли это выражение в некоторой монографии или специальном справочнике, то не забудьте сделать ссылку в виде свободного текстового блока на диаграмме:



К сожалению, г-н Rande Johnson не владеет русским языком и комментарии в примерах сделаны на его "рідній мові". Но Вы-то можете делать заметки на том языке, который Вам нравится, даже на китайском. Не забывайте об этом!

Будете следовать правилу ***№5*** - получите возможность автоматически создать отчетную документацию о выполненной работе, используя Документационные Инструментальные средства LabVIEW. Если не хотите делать комментарии - не делайте, но отчет писать все равно придется. А это - время, а время - деньги. Вот так! Решайте сами.

6 Используйте нормальные описательные имена

Если Ваша программа/подпрограмма предназначена для преобразования значений кода напряжения в инженерные единицы или сохранение данных в виде таблицы, то так и назовите эти VI: Convert Voltages to Engineering Units.vi и Save Data Spreadsheet File.vi. И не используйте нечто типа VI # 1.vi или Untitled.vi. Для подпрограмм, которые что-то выполняют, подойдут слова Open, Close, Save, Calculate, Update и т.п., а для пользовательского интерфейса - Display, Panel, View, Screen. Неплохо использовать префиксы к именам VI, указывающих категорию или функциональность. Это просто необходимо, если Ваши VI помещаются в библиотеку .llb: процедуры группировки, перемещения, классификации упрощаются.

На этом с общими правилами можно и закончить. *Переходим к рекомендациям, которых следует придерживаться при создании передних панелей и интерфейсов пользователя. Прежде чем Вы начнете что-либо делать с Передней Панелью, решите - это обычный VI, или панель оператора, или диалоговое окно. Каждый из этих VI имеет свою специфику.*

7 Обычным VI – базовые элементы управления

Пользуйтесь стандартной палитрой управляющих элементов ***Controls*** для простых VI, которые нужны только программисту. Не беспокойтесь об особой "красоте" - время дорого, лучше выполните все аккуратно и используйте мелкий шрифт для имен.

8 Группируйте логически однотипные управляющие элементы

Если у Вас есть управляющие элементы на контрольной панели, которые связаны друг с другом, сгруппируйте их, используя кластер, или поместите их на декоративную панель ***(Decoration)***. Например, результаты измерений и вычислений используются совместно - примените кластер, а если не используются - тогда, декоративный бокс, на который помещаются индикаторы.

9 Размер окна

Убедитесь в том, что ваша панель будет отображаться на экране монитора целиком. Если Вы не уверены, какой размер экрана выбрать, например 640X480 или 800X600 по умолчанию, тогда можно использовать ***Screen sizing*** функции LabVIEW.

10 Не устанавливайте управляющие элементы слишком близко

Придерживаться этого правила следует особенно в случаях использования сенсорных экранов TouchScreen или управления режимами клавишой Tab. Делайте вещи простыми на вид и доступными для пальцев. Попробуйте нарисовать серые или белые линии между контрольными элементами на передней панели. Если Вы используете TouchScreen, то минимально допустимым считается элемент площадью 1.2 кв. дюйма (около 3 кв.см). Если элемент меньше или несколько элементов стоят слишком близко друг к другу, Вам будет трудно на них позиционировать курсор и нажимать кнопку, не задевая соседнюю.

11 Не размещайте кнопки диалога друг поверх друга

Скажем, Вы имеете диалог, в котором спрашивается, хочет ли пользователь переименовать или заменить файл. Предположим, он выбрал "Заменить" (replace) и открывается другое диалоговое окно, в котором спрашивается "Вы уверены, что хотите переименовать файл?". Если Вы поместите кнопку "Заменить" на тоже место, что и кнопки "Отменить" и "О.К.", относящиеся к следующему диалогу, будет очень просто нажать не на ту кнопку ненамеренно. И вы будете считать, что сделали то, что надо, а на самом-то деле - нет.

12 Используйте клавиатуру или навигацию с помощью клавиатуры для выбора по умолчанию

Если Вы хотите сохранить или применить функцию "по умолчанию " (default), когда оператор нажимает клавишу Esc, используйте атрибут ***KeyFocus*** или ***Key Navigation***, установленный в режим ***write***. Это особенно важно для рискованных операций, таких как перезапись файла, когда Вы не хотите, чтобы оператор сделал что-то, что ему не следует делать. Вы можете этого добиться очень просто ...

13 Специальные слова для имен кнопок

Вместо OK или Cancel лучше используйте для имен кнопок такие слова, как "Запись" (Save), "Замена"

(Replace), "Прекратить" (Quit) и т.п. Это сделает программу значительно проще для оператора, хотя и несколько усложнит Вам "жизнь".

14 Всегда включайте опции Cancel и Back

Если пользователь знает, что может нажать что-то на экране без возникновения в дальнейшем проблем, он чувствует себя комфортно: экспериментирует, самообучается, уверен в безопасности и всегда может вернуться назад. Помните Undo?

Домучив предыдущую страницу, вы быстренько ее перевернули и ... Думали нравоучения закончились? Нет уж, они только начинаются - "Надо Федя, надо. Для пользы дела". Но если Вы немножко все-таки устали, организуйте себе кофебрейк или просто подышите свежим воздухом, и давайте продолжим.

15 Имена элементов типа Boolens должны отражать, что имеется в виду под состоянием "Вкл"

Лаконично и ясно сформулируйте, что же делает элемент "Выключатель", когда находиться в состоянии "Вкл". Например, Reset, Initialize, Cancel - все ясно и понятно, что произойдет если нажать на клавишу. Постарайтесь избегать таких имен как "НЕ отображать диалог" ("Don't display dialog") или "Не заменять". Можно также рекомендовать использовать вопросительные имена, например "сканировать?" вместо "сканирование".

16 Выставляйте Управляющие элементы и Индикаторы в положение, в котором они должны быть на коннекторе (Show Connector)

Старайтесь размещать входы слева, а выходы справа на панели коннектора. Панель коннектора находится в правом верхнем углу передней панели и открывается, если щелкнуть правой кнопкой мыши и выбрать пункт ***Show Connector***. Размещайте Ваши идентификаторы задачи ***taskID*** и ссылки ***refnums*** соответственно слева вверху и справа вверху на коннекторе, а кластеры ошибок ***ERROR*** слева и справа внизу. Эти замечания естественно касаются VI, не содержащих интерфейса пользователя. Для VI же с диалоговыми окнами логика управления, простота и интуитивность являются более значимыми и это правило можно проигнорировать.

17 Пишите на передней панели в скобках значения по умолчанию

Если управляющие элементы подпрограммы используют значения по умолчанию, конечно же эти значения должны присутствовать на экране. Например, используется вход Reset, и Вы пометили его как Reset (F), показывая, что Reset не может быть вызван до тех пор, пока на этом входе не будет установлено значение True. Если значение по умолчанию не очевидно, используйте более подробное описание, например open mode (0:read/write). А если единицы измерения важны, то Вы должны и это указать, пример - Задержка (500мс).

18 Коннекторы

Один раз определитесь с видом разъема и работайте с ним, например выбирете 12 контактный terminal для всех своих VI. Использование однотипного коннектора позволяет очень просто соединять между собой иконки Ваших программ. Вы можете добавлять и удалять входы и выходы без необходимости перелинковки подпрограмм. В противном случае могут возникать соединения не к тем выводам разъема. Если необходимо большее количество выводов, объедините часть входов в кластер. Возможно также конструировать VI из некоторого количества SubVI, каждый из которых имеет небольшое количество ножек разъема.

19 Используйте "обязательные" (required), "рекомендованные" (recommended) и "опциональные" (optional) установки для выводов коннектора

Это можно сделать, если выбрать пункт ***This Connector is..*** Если Вы работаете с VI, "вход" которого должен быть обязательно подсоединен, следует выбрать пункт ***This Connection is Required.*** Используя этот прием, Вы сэкономите немало времени при отладке. Аналогично, если Вы не собираетесь изменять параметры на соответствующем выводе, укажите ***Optional***. В стандартном окне ***Help*** такой терминал будет серым. Кроме того, Используйте маленькие буквы (Small Fonts) в именах иконок. Это очень удобный шрифт. Дважды кликните мышкой на букве ***A*** в окне редактирования иконки и выберите маленький шрифт.

20 Красивые графические иконки

Программу, состоящую из одинаково выглядящих иконок подпрограмм, отлаживать очень трудно, да и не очень-то приятно. Никакого удовольствия.

21 Шрифт и Цвет

Для создания паленей и диаграмм в приложениях (applications) старайтесь использовать встроенные шрифты. Это касается и цветов. А для оператора можно просто создать конфигурационный файл. Вы потратите на это время, но сделать это нужно только один раз.

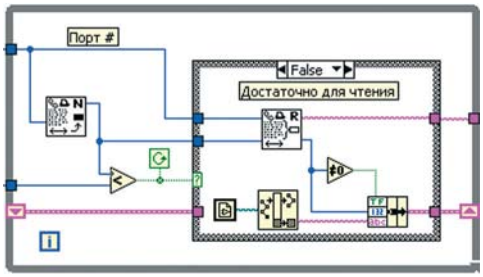
от конструирования интерфейса пользователя на передних панелях перейдем к блок диаграмме, и начнем с подготовки среды LabVIEW

22 Создание палитр клиентских меню

Выполните перегруппировку палитр, установленных по умолчанию, выделяя наиболее для Вас важные части. Например, можно рекомендовать разместить loop-структуры, file I/O и некоторые утилиты слева вверху. Они часто используются. Меньшее количество нажатий на клавишу мыши действительно ускоряет процесс программирования.

23 Добавьте функцию ожидания (wait) во все Интерфейсы Пользователя и интенсивно выполняемые циклы

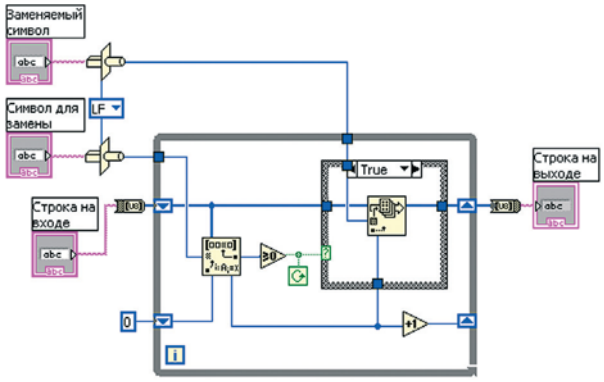
Если Вы решите выбрать только одно правило из всех рассмотренных на этом уроке и следовать ему, то выберите ЭТО ПРАВИЛО. Это самое лучшее из всех. Вы будете приятно удивлены, насколько лучше Ваша программа будет работать, если вставить небольшую задержку в каждый цикл пользовательского интерфейса. Некоторые циклы в Вашей программе могут работать так быстро, как только возможно, "уморив" все Ваши остальные циклы. Отсутствие задержки приводит к пустой трате времени процессора и "убивает" другие части программы. А добавляя небольшую задержку, Вы позволяете LabVIEW и операционной системе освободить процессор для выполнения чего-нибудь другого. Кроме того, оператор редко может видеть или реагировать на события чаще, чем через 0.1 сек. Так что задержка в 100 мс не будет заметной и улучшить исполнение программы. Это правило следует учитывать и для циклов, не связанных с пользовательским интерфейсом, но которые интенсивно расходуют время. Хороший пример - обычное чтение байтов из сериального порта. Вместо того, чтобы ставить задержку в цикле, в котором читаются значения, поставьте маленькую задержку после каждой проверки счетчика байтов. Это более предпочтительный вариант. В этом случае, если все байты уже считаны во время работы программы, никакого ожидания возникать не будет. А если по какой-то причине данные не считаны, в программе возникнет ожидание и чтение повторится. Но в этот короткий промежуток времени LabVIEW обеспечит Вам выполнение чего-нибудь еще.



24 Технология эффективного использования памяти для массивов и манипуляций со строками

Если вы действительно хотите, чтобы Ваша LabVIEW-программа работала "со свистом", внимательно прочтите и запомните концепцию, изложенную в руководстве по LabVIEW - ***Memory Usage***. Вот что нужно делать для экономии памяти. Оставляйте функции ***Build Array*** и ***Concatenate String*** вне цикла. Пытайтесь использовать ***Replace Array Element*** везде, где только возможно. Если Вы получаете данные в результате большого количества повторов цикла и программа должна работать быстро, выделите сегмент в памяти (pre-allocate), используя ***Initiate Array*** перед циклом, затем используйте ***Replace Array Element*** в цикле и далее ***Reshape Array*** после окончания работы цикла, если это необходимо. ***Reshape Array*** устанавливает правильный размер массива.

Для больших строк рассмотрим помещение строки в массив байтов и использование функций массивов вместо функций строк. Это можно выполнить как утилиту для изменения символов окончания строки и тогда Ваш VI будет работает в десять раз быстрее, чем при использовании функций строк:



Использование большого массива или строковых индикаторов на передней панели нужно делать, если только это действительно нужно. Каждый индикатор на передней панели делает копию данных, имеющихся в блок диаграмме. Т.е. Вы имеете две копии одних и тех же данных. Это не критично для небольших индикаторов или маленьких массивов, но просто разорительно для больших структур данных.

25 Ограничьте использование локальных и глобальных переменных

Об этом свидетельствует опыт многих LabVIEW программистов. Безусловно, и глобальные и локальные переменные абсолютно необходимы при программировании в LabVIEW. Но начинающий программист должен обходить их дальней дорогой, ибо, поверьте, неизбежно возникнут проблемы, связанные с временем исполнения программы.

В представленном ниже примере, данные не связаны между собой, поэтому глобальная переменная может быть считана прежде, чем в нее будут записаны данные, а в результате будет получен неправильный результат.
Оптимизация скорости исполнения возможна, но это не так просто. Со временем, когда у Вас будет больше опыта, Вы будете лучше понимать, когда и где нужно использовать локальные и глобальные переменные.

Другой недостаток в использовании локальных и глобальных это то, что они "кушают" память, а именно создаются копии данных в памяти. Т.е. если у Вас на диаграмме присутствует, например, индикатор и локальная переменная от него, то Вы получите удвоение размера используемой памяти. С малыми массивами данных проблем нет, но большие структуры - одна головная боль.

Будьте очень осторожны с глобальными переменными, если Вы запускаете несколько циклов параллельно или используете многопотоковость. В этом случае очень трудно или практически невозможно управлять синхронизацией. Лучше используйте такие функции синхронизации LabVIEW, как ***Semaphore***, ***Notifier***, ***Queue***, ***Randesvous***.

Нельзя не согласиться с тем, что использование глобальных и локальных переменных экономят время разработки программы и упрощают сам процесс программирования. Но если у Вас есть желание "вылизать" проект, следует заменить их на альтернативное решение.

26 Избегайте использования последовательности во всей полноте, но с одним исключением

Последовательные структуры плохо выглядят. Многие программисты ворчат о том, что последователь-

ные структуры должны категорически быть удалены из LabVIEW. Они используются для управления "данными" и программирования потоков в LabVIEW, и это действительно необходимо в настоящее время. Почти во всех функциях ввода/вывода в LabVIEW сейчас используются кластеры ошибок, которые просто соединяют в последовательную цепочку, без необходимости использования ***Sequence***. Таким же образом параллельные процессы могут использовать предпочтительные функции синхронизации, такие как ***Semaphores*** и ***Notifiers***, взамен последовательной структуры для управления ходом выполнения программы и пересылки данных. LabVIEW-программисты не любят ***Sequence*** еще и потому, что они скрывают часть кода программы во фреймах, помимо того, который виден в данный момент. Хочется видеть всю программу на экране.

Но нужно сказать, одиночный фрейм структуры ***Sequence*** по прежнему важен: он ничего не скрывает, дает Вам уверенность, что нечто исполняется прежде, чем будет исполнен код во фрейме. Например, Вы можете захотеть инициализировать все управляющие элементы прежде, чем начнет исполняться программа или откроется нужная панель.

27 *Ограничьте диаграмму одной страницей*

Стиль LabVIEW-программиста - расположить все subVI на одной странице, избегая скроллинга. Что это Вам дает? Первое - создается модульность и VI выглядит достаточно просто. Это очень, и даже очень хорошо! Второе - Вы видите всю программу сразу, не делая скроллинг страницы для поиска где, что и с чем соединено.

Давайте организуем еще один перерыв, на этот раз - активный, можно с кофе или чаем. И оставим на время полезные советы, а займемся техникой отладки VI.

Поиск ошибок

Когда VI содержит ошибки или не выполняется, на кнопке ***Run*** появляется сломанная стрелка. Чтобы просмотреть ошибки, нажмите на сломанную стрелку и откроется ***Error List*** (Перечень ошибок). Выберите одну из перечисленных ошибок и затем нажмите на ***Show Error*** (Показать ошибку) или кликнете двойным щелчком мыши. Далее откроется блок диаграмма, на которой будет мигать "неисправный элемент". В окне ***Error List*** в разделе ***Details*** (Подробно) содержатся краткие сведения об ошибке.

Трассировка

Если Вы хотите визуально проследить, что происходит в каналах или в какой последовательности выполняется ваша программа, нажмите на кнопку ***Execution Highlighting*** (Подсветка выполнения) и запустите VI. Символ изменится и вы будете наблюдать как данные проходят по VI.

Пошаговое выполнение

Во время отладки, Вы можете пошагово выполнить всю программу. Чтобы активировать пошаговый режим, нажмите на паузу и запустите программу. Первый узел на блок диаграмме начнет мигать. Теперь нажмите на кнопку ***Step Into*** (Шаг внутрь) или ***Step Over*** (Шаг через) и программа перейдет к следующему узлу.

Разница между этими возможностями заключается в том, что ***Step Into*** при выполнении узла, будет открывать его блок диаграмму в новом окне и продолжать пошаговое выполнение там. Для завершения пошагового выполнения нажмите на кнопку ***Step Out*** .

Показания пробника

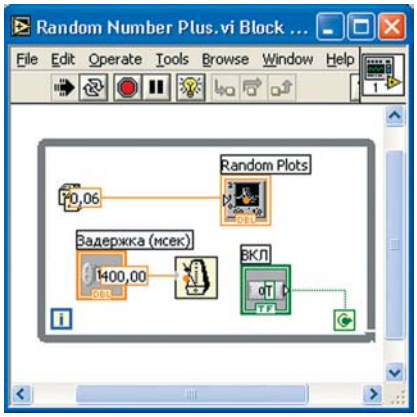
Иногда нужно постоянно быть в курсе того, что происходит в канале данных. Для этого в ***Tool Palette*** (Палитра инструментов) находится элемент под названием ***Probe Data*** (Опробовать данные) . Нажимаем на него и устанавливаем на нужный канал. Рядом появится окошко с индикатором, соответствующего типа данных.

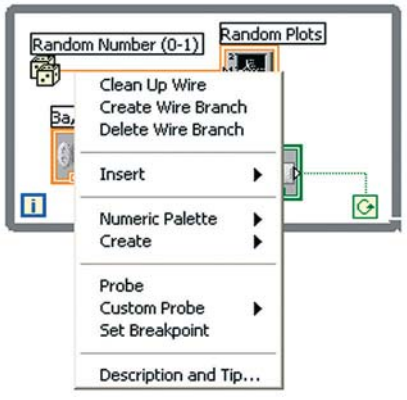
Контрольные точки

Выполнение программы можно остановить в определенных Вами контрольных точках, например в подпрограмме, на проводнике и т.п. Необходимый для этого инструмент ***Set/Clear Breakpoint*** (Установить/Убрать контрольную точку) находится в палитре инструментов. Выберите его и нажмите на любой элемент, где Вы хотите устанавливать или убрать контрольную точку. Контрольные точки изображены в виде красных рамок диаграмм, и красных точек для проводов. После запуска VI, дойдя до контрольной точки, активирует паузу.

*А теперь - практическая работа с использованием трассировки, контрольных точек, пошагового исполнения и пробников. Создаете VI как показано ниже и назовите его **Random Number Plus.vi**.*

В окне диаграмм нажмите на кнопку ***Execution Highlighting*** на панели инструментов, включите переключатель "ВКЛ" и нажмите кнопку Run. Обратите внимание, что точки, “бегущие по проводам” показывают Вам поток данных и порядок выполнения subVI.

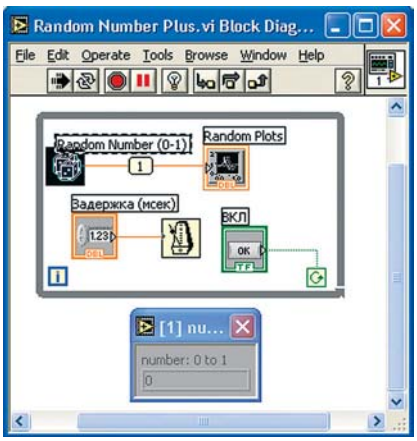




Снимите выделение кнопки ***Execution Highlighting***, и нажмите на переключатель "ВКЛ" для остановки VI. Откройте всплывающее меню на проводнике между функцией ***Random number (0-1)*** и терминалом ***Random Plot***, и выберите ***Probe***.

Установите контрольную точку на функции ***Random number (0-1)***. Функция будет иметь красное обрамление.

Включите переключатель "ВКЛ", и запустите



VI. На диаграмме будет вспыхивать функция ***Random number (0–1)***, указывая, что она готова к выполнению.

Нажмите на кнопку ***Step Over*** для того, чтобы выполнилась функция ***Random number (0–1)***. Сгенерированное значение теперь отображается в окне пробника.

Нажмите на кнопки ***Step Into*** и ***Step Over.*** Вы увидите, как Работает пошаговый режим в LabVIEW. Для того, чтобы вернуться в обычный режим, просто нажмите на кнопку ***Pause***.

Выберите из числа своих VI примерчик посложнее и попросите коллегу внести в него "некоторые коррективы". Не трудно догадаться, какие именно. А затем, используя описанную технику отладки, найдите эти "улучшизмы". Но заняться этим следует после того, как вы пройдете этот урок до конца. Осталось чуть-чуть.

Вернемся к полезным советам г-на Johnson и рассмотрим Правила Соединений (Wiring Guidelines) функциональных блоков на диаграмме с помощью инструмента ***wiring tool****.*

28 Выполняйте соединение слева направо и не делайте соединения невидимыми

Помните правило, в котором говорится о коннекторах для подсоединения к VI? Как раз тот самый случай. Если Ваши коннекторные панели сконструированы одинаково, то все соединения делаются одним щелчком. Выход установленный справа непосредственно и прямо соединяется с входом следующего блока. А если коннекторные панели имеют одно и то же число терминалов, то и не надо изгибать проводники для соединений между ними.

Никогда не прокладывайте проводники один под другим! Если это сделать, Вы больше не сможете видеть - выполнены все соединения или нет. К тому же Вы можете и забыть о том, чего не видите.

Могут возникнуть большие проблемы, если Вы выделите некоторое количество точек в цикле и соедините их так, что изображение цикла наложится на проводники. В этом случае проводники не будут видны, и Вы можете подумать, что эти проводники не участвуют в работе цикла.

Если Вы удаляете или перемещаете часть программы с невидимыми линиями, то будут удаляться и невидимые линии, а что будет после этого - неизвестно. Если Вам повезет, то увидите предупреждение LabVIEW, если нет - то...

29 Remove Bad Wires – удобный инструмент, но

Избегайте использовать ***Remove Bad Wires***. Этот инструмент удаляет любые неправильные соединения и не только те, на которые Вы смотрите, например те соединения, которые могут быть частью незаконченной программы. Если Вы удалите все, то потом нужно будет возвращаться назад и восстанавливать соединения в части программы, которую Вы не закончили. Некоторые соединения могут быть удалены, и при этом не будет вызова ошибки. Если Вам доводилось программировать на C или Basic, то представьте себе, что Ваш редактор сразу удалил все строки с синтаксическими ошибками. Это будет катастрофа, не так ли?

Используйте ***Remove Bad Wires*** тогда, когда у Вас небольшая программа и возникло одно или два "плохих" соединений. Вы абсолютно уверены, что ничего не испортите? Тогда нажимайте ***Ctr–B***.

30 Используйте тройной "клик" для просмотра ошибки соединения

У вас возникли проблемы с соединением? Как их решить без применения ***Remove Bad Wires***? Попробуйте щелкнуть мышью по плохому соединению. Одиночный щелчок выделит сегмент соединения, двойной - от точки до точки, тройной - всю линию. Тройной щелчок специально используют для поиска всех взаимосвязанных терминалов, тех о которых Вы знаете и не знаете. И нет другого пути, когда соединены два входных терминала (а этого нельзя делать) и один из проводников спрятан под иконкой. И если Вы будете использовать двойной щелчок и потом удалите все, то Вы удалите и правильную часть соединения. В такой ситуации следует трижды щелкнуть, чтобы увидеть все точки подключения. Нет более простого и правильного пути для того, чтобы быть уверенным в правильности соединения, чем тройной щелчок.

31 Используйте Create Control/Constant так часто, как можете

Еще со времен LabVIEW 4.0 стало возможным создавать константы на диаграмме простым щелчком по правой клавише мыши и выбором пункта Create Constant. Помните об этой возможности и со многими неприятными ошибками Вам не придется иметь дело. Пусть, например, в подпрограмме используется Enumerate Constant и для управления Case используется некоторое число, которое спустя некоторое время изменится, то Вы и не будете осведомлены о том, что случилось. Лучше создайте константу и тогда, если enumerated constant когда-нибудь изменится, Вы сразу получите сообщение об ошибке. Подумайте о часах, потраченных на поиск ошибок, которые может сохранить Вам этот прием.

"Чистая" диаграмма свободна от ошибок, выглядит более профессионально, проста в сопровождении и была написана кем-то, кто сначала подумал, а потом начал все соединять. "Грязная" же диаграмма обычно полна неприятных неожиданных ошибок, чувствуется, что написана очень быстро и непродуманно. Всего несколько минут, потраченных на "причесывание" диаграммы, сэкономят часы при отладке! Совсем без ошибок не бывает. В любой программе будут встречаться ошибки. Но в хорошо и профессионально написанных приложениях возни с ними будет не много.

32 Используйте кластеры Error In и Error Out

Во всех VI, в которых могут возникнуть проблемы во время исполнения, используйте кластеры ***Error In*** и ***Error Out*** из палитры ***Array and Cluster Control***. Это обычная практика для VI, работающих с файлами, серальным портом, GPIB (приборный интерфейс) и т.п. Названные кластеры обеспечивают простой и последовательный путь для отчета об ошибках, возникших во время вызова VI. Они также могут быть использованы и как вызов для управления ходом выполнения программы:

В этом примере ***Open File*** должен исполняться перед ***Read File*** и ***Close File*** после того, как все будет выполнено. Заметьте, как это выглядит. Никаких структур ***Sequence***, которые скрывают код, никаких промежуточных проверок ошибок. Если ошибка возникнет до исполнения этой части, она будет обработана в этом кластере немедленно.

33 Используйте NaN (Not-a-Number) вместо кластеров Error, где это уместно

Некоторые VI, особенно математические функции, не выдают информацию в кластер ***Error***. Вместо этого, если VI не исполняется, на его выходе устанавливается ***NaN***. В этом есть несколько преимуществ. Первое - любой вызванный VI может использовать ***Not a Number/Path/Refnum*** для того, чтобы посмотреть успешно ли исполняется функция. Второе - ***NaN*** не отображается на графике. Так что, что когда формируется график, а Ваши данные вдруг вышли за границы, то на нем не будут отображены данные, вышедшие за границы.

34 Не спешите смотреть на ошибки – пропустите их дальше по линии

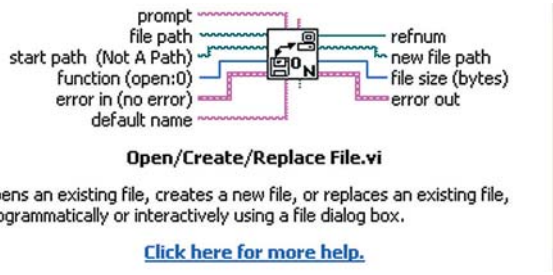
Займитесь анализом ошибок в соответствующее время. Например, в примере, приведенном выше, нет необходимости отслеживать возникновение ошибки в каждой точке. Вместо этого подождите, пока сообщение об ошибке появится в конце.

35 Передавайте ошибку без изменений

Если ошибка обнаружена до Вашего VI , пропустите исполнение Вашей программы и передайте ошибку нетронутой. Не делайте дополнительного переписывания сообщения об источнике или коде ошибки, или Вы никогда точно не будете знать, что произошло.

36 Никогда не делайте stop или не показывайте много диалоговых окон

Error Cluster не только позволяет разместить сообщение в одном месте, но также исключает необходимость все время нажимать на ***OK*** или ***Cancel***. LabVIEW-профессионалам нравится использовать ***LabVIEW Queue*** для управления сообщениями об ошибках. Создается ***queue*** (очередь) в главной программе и запускается ***Remove Queue Element*** в отдельном цикле. Все subVI пересылают статусы ошибок или другую инфомацию (NaN) в главную программу, используя ***Insert Queue Element***. Это не только простой, но и очень эффективный способ. Одно предупреждение: управление ошибками программно приводит к необходимости выключать большое количества информационных диалогов, которые будут появляться из большого количества встроенных VI. Не все VI имеют эту опцию, так что Вам может понадобится модифицировать их.



Управление ошибками в центральном окне может потребовать выключение ***advisory dialog***, встроенный в VI.

37 Создавайте свои коды ошибок и используйте их в дополнении к кодам ошибок LabVIEW

В большинстве случаев Вы будете обнаруживать подходящие коды ошибок в ***Error Ring*** в секции ***Addition Numeric Constants*** в палитре ***Numeric***. Но это не значит, что Вы ограничены только этими кодами. LabVIEW допускает значительно больше сообщений об ошибках - Вы можете определить их сами в диапазоне 5000..9999. Используйте ***General Error Handler*** из палитры ***Time&Dialog***. Если Вы попрежнему не можете найти подходящий код ошибки, создайте свой собственный в диапазоне 5000..9999.

38 Используйте отрицательные значения кодов ошибок для обозначения фатальной ошибки

В соответствии с Соглашениями LabVIEW, если Вы не можете завершить Вашу задачу, установите статус кода Вашей ошибки отрицательным. А если задача завершена, но что-то произошло, установите значения кода положительным. Потом можно будет разобраться, что же произошло.

Все. Рекомендации закончились, а вместе с ними, наверное, и Ваши мучения. Ура! Но! Если Вы хотите, чтобы сам процесс программирования в LabVIEW стал для Вас райским наслаждением, перечитайте наставления этого урока еще раз. :-)

"Быстрая разработка, легкость сопровождения и многократность использования подпрограмм - вот результат использования правил 1..38", М. И. Манжелий, Инженерный Центр "Автоматизированные системы контроля", Москва, Россия.